XPS-Q8

Universal High-Performance Motion Controller/Driver

Configuration Wizard Documentation

V1.4.x

Newport®
Experience | Solutions

Precision Motion – ***Guaranteed***™

Preface

Confidentiality & Proprietary Rights

©2015 Newport Corporation
1791 Deere Ave.
Irvine, CA 92606, USA
(949) 863-3144

Table of Contents

XPS
Universal High-Performance Motion Controller/Driver

1.0 Introduction

This manual refers to the configuration of the XPS motion controller/driver to motors and stages that are not included in the XPS general stage database, e.g. non-Newport stages. The manual will present all possible configurations of the XPS controller with regards to drive and control capabilities.

The XPS controller uses two configuration files, named "system.ini" and "stages.ini". The files are located in the "..\admin\config" folder of the XPS controller. These configuration files are read during the booting of the controller. The "system.ini" file specifies the system configuration and the configured motion groups. The "stages.ini" file defines the parameters for all positioners. The aim of this manual is to provide a better understanding of the drive and control capabilities of the XPS controller and possible settings in the "stages.ini" file. It is important that users have a working understanding of the structure of this file, because it may be necessary to make modifications to the default parameters to access all features of the XPS controller. In order to configure the XPS controller to drive non-Newport stages, it is important that users have an in-depth understanding of this file and the meaning of the included entries.

This manual is presented in the same sequence as users would enter data in the pages of the configuration wizard web site tool. However, this order might not be most intuitive. It is therefore recommended that users first have a good understanding of **all** required settings before beginning any configuration.

In each section we provide a detailed definition of each parameter, the physical meaning and one method to set it. Due to the many potential options for configuration, we can not detail all strategies and methods. Especially for the definition of the tuning parameters (PID, filters, etc.). Please refer to supporting literature for an in-depth treatment of tuning.

IMPORTANT NOTE ABOUT THE UNITS

The XPS controller accepts any dimension for the position unit such as: mm, inch, µm, deg, rad, etc. In this documentation, we use the generic term "unit" for the position unit. This generic unit is carried forward into units that reference the position unit, for example speed and acceleration would carry units such as: units/s or units/s². The physical dimension assignment of the position unit for closed-loop systems is done by *stage displacement per encoder count* as part of the parameters of the *position encoder interface.* For open-loop systems the physical dimension assignment is done as part of the parameter settings for the driver command interface examples include: *stage displacement per motor full step* or *command voltage at minimum target position.* It is important to note that the position unit in the configuration files will determine the values of all derived parameters. It is important to note, as the choice of the position unit will impact most parameters.

Index on stage.ini File Parameters

To improve understanding, the names of parameters used in the website configuration tool are more descriptive and therefore slightly different from the names used in the XPS stages.ini configuration file. Thus, in each section we detail the names used by the website configuration tool and the names used in the stages.ini configuration file.

At the end of this document we provide an index of the stages.ini file entries. Please use this index when searching for documentation on specific entries in the stages.ini file.

2.0 Web Site Description

The integrated assistant tool *STAGE, Manual construction menu*, accessible when logged in as administrator, has been designed by Newport to help users configure the XPS controller for motors and stages that are not included in the XPS general stage data base, e.g. stages not manufactured by Newport. The tool generates a new entry in the customer's stage database, *stages.ini*, which is located in the "..\admin\config" folder of the XPS controller.

To generate a new stage entry using the *manual construction menu*, two levels of settings are defined.

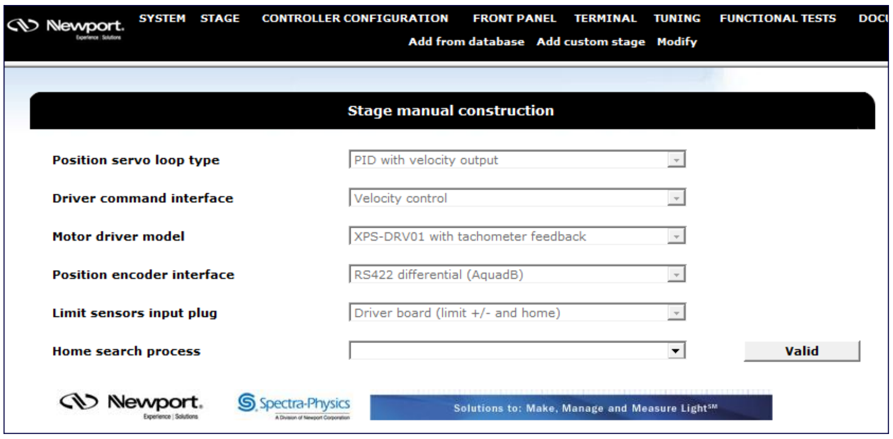
- 1st level settings

These 6 settings define the skeleton of the configuration. They must be set in a fixed sequence. The software updates dynamically and lists only those next options that are compatible with the previous selections. This avoids configurations that are not supported by the XPS controller. Once any of these settings is completed,it can no longer be changed.

- 2nd level settings

There are numerous parameters to set for each 1st level setting. These 2nd level settings can be done in any order and also modified later. However, a configuration can not be completed until all settings are complete.

<u>1st Level Settings</u>



Stage manual construction

Position servo loop type	PID with velocity output
Driver command interface	Velocity control
Motor driver model	XPS-DRV01 with tachometer feedback
Position encoder interface	RS422 differential (AquadB)
Limit sensors input plug	Driver board (limit +/- and home)
Home search process	

Figure 1: 1st level settings of the XPS manual stage construction tool.

The 1st level settings outlined above define the skeleton for the configuration (see Figure 1). These settings must be set in the same order as listed, which means starting with the *Position servo loop type*, followed by *Driver command interface*, and so on. These settings are presented in detail in chapters 3 and 4. The scroll down list for any of the six settings appears only after all previous settings are done. For instance, the *motor drive model* can only be specified when the *position servo loop type* and the *driver command interface* have been defined. The software dynamically updates the scroll down lists based on prior selections, to offer the correct settings for the determined configuration. Only those options that are compatible with the previous choices are presented.. This avoids configurations that are not supported by the XPS controller. However, this tool cannot compensate for configurations that are not compatible with the connected hardware, e.g. stages and external amplifiers (when using XPS-DRV00 drive).

Once any of the 1st level settings has been set, it can no longer be changed. To abort the current configuration, click on the manual construction menu again.

When all six 1st level settings are done, click on Valid to apply the current choices. Then, the 2nd level settings become available (see Figure 2).

The type and sequence of these 1st level settings may not be intuitive for all users, however this methodology is the only option that provides full access to all control capabilities of the XPS controller while providing on-line updates to eliminate impossible configurations.

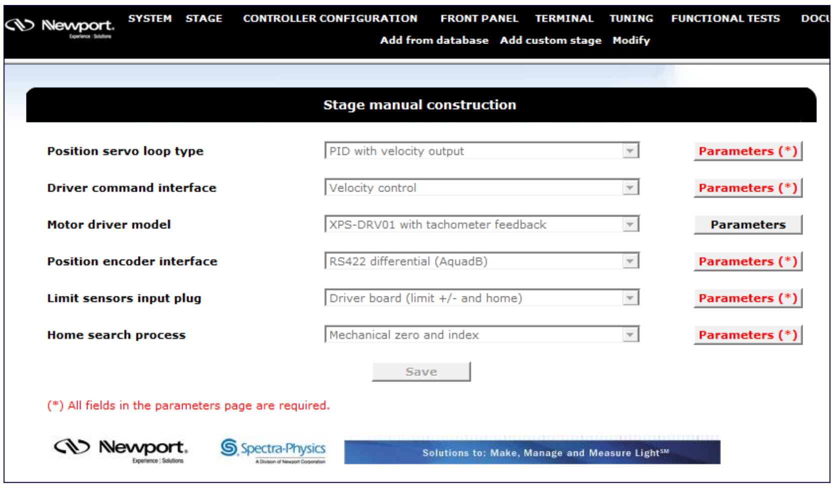
The table below presents position servo loop type with driver command interface and motor driver model according to stage configuration. This table presents the most common motor types and feedback systems, but nevertheless is only a fraction of the total drive and control capabilities of the XPS, and should not be considered exhaustive.

Stage configuration	**Position servo loop type**	**Driver command interface**	**Motor driver model**
DC motor <= 3A with encoder and tachometer	PID with velocity output	Velocity control	XPS-DRV01 with tachometer feedback
DC motor <= 3A with encoder, no tachometer	PID with motor voltage output	Voltage control	XPS-DRV01 without tachometer feedback
DC motor <= 5A with encoder and tachometer	PID with velocity output	Velocity control	XPS-DRV03 with tachometer feedback
DC motor <= 5A with encoder, no tachometer	PID with acceleration output	Acceleration control	XPS-DRV03 for acceleration control
	PID with motor voltage output	Voltage control	XPS-DRV03 for voltage control
DC motor <= 1.58A with encoder and tachometer	PID with velocity output	Velocity control	XPS-DRV03H with tachometer feedback
DC motor <= 1.58A with encoder, no tachometer	PID with acceleration output	Acceleration control	XPS-DRV03H for acceleration control
	PID with motor voltage output	Voltage control	XPS-DRV03H for voltage control
DC motor > 5A with encoder and tachometer	PID with velocity output	Velocity control	XPS-DRV00/XPS-DRV00P for external driver
DC motor > 5A with encoder, no tachometer	PID with acceleration output	Acceleration control	XPS-DRV00/XPS-DRV00P for external driver
	PID with motor voltage output	Voltage control	XPS-DRV00/XPS-DRV00P for external driver

Stage configuration	**Position servo loop type**	**Driver command interface**	Motor driver model
Stepper motor <= 3A with encoder	PI with position output	Sine/cosine position control	XPS-DRV01 for stepper motor
Stepper motor <= 3A without encoder	No servo loop with position output	Sine/cosine position control	XPS-DRV01 for stepper motor
Stepper motor > 3A with encoder	PI with position output	Sine/cosine position control	XPS-DRV00/XPS-DRV00P for external driver
Stepper motor > 3A without encoder	No servo loop with position output	Sine/cosine position control	XPS-DRV00/XPS-DRV00P for external driver
Stepper motor > 3A with encoder	PI with position output	Pulse/Direction or Pulse+/Pulse- position control	XPS-DRV00P for external driver
Stepper motor > 3A without encoder	No servo loop with position output	Pulse/Direction or Pulse+/Pulse- position control	XPS-DRV00P for external driver
Linear/brushless motor <= 5A, double command input 120° UV phase driver	PID with acceleration output	120 deg UV phase acceleration control	XPS-DRV02
Linear/brushless motor <= 7A, double command input 120° UV phase driver	PID with acceleration output	120 deg UV phase acceleration control	XPS-DRV02P
Linear/brushless motor <= 5A, single command input driver	PID with acceleration output	Acceleration control	XPS-DRV00/XPS-DRV00P for external driver
Linear/brushless motor <= 5A, double command input (60° or 90° UV phase) driver	PID with acceleration output	Acceleration control - 60 deg UV phase - 90 deg UV phase	XPS-DRV00/XPS-DRV00P for external driver
Linear/brushless motor > 5A	PID with acceleration output	Acceleration control - 60 deg UV phase - 90 deg UV phase - 120 deg UV phase	XPS-DRV00/XPS-DRV00P for external driver

Table 1: XPS 1st level settings for the most common motor types and feedback systems

2nd Level Settings



Stage manual construction

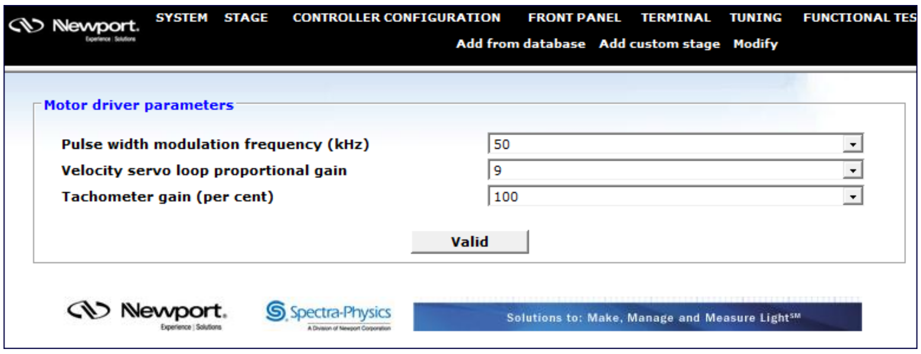
Position servo loop type	PID with velocity output	Parameters (*)
Driver command interface	Velocity control	Parameters (*)
Motor driver model	XPS-DRV01 with tachometer feedback	Parameters
Position encoder interface	RS422 differential (AquadB)	Parameters (*)
Limit sensors input plug	Driver board (limit +/- and home)	Parameters (*)
Home search process	Mechanical zero and index	Parameters (*)

Save

(*) All fields in the parameters page are required.

Figure 2: 2nd level settings of the XPS manual stage construction tool

The 2nd level settings define all details of the configuration. These settings can be done in any order and changed at any time. To access the 2nd level settings, click on the *parameters** button. A new page opens prompting the settings for the parameter setting (see Figure 3 and Figure 4 for example). The type and number of settings in each of these screens will depend on the first level setting. Define all settings and press the valid button to apply them. This gets you back to the previous page. When all parameters are set in any of the sub-pages the font color of the parameters button changes from red to black and the asterisk (*) disappears.

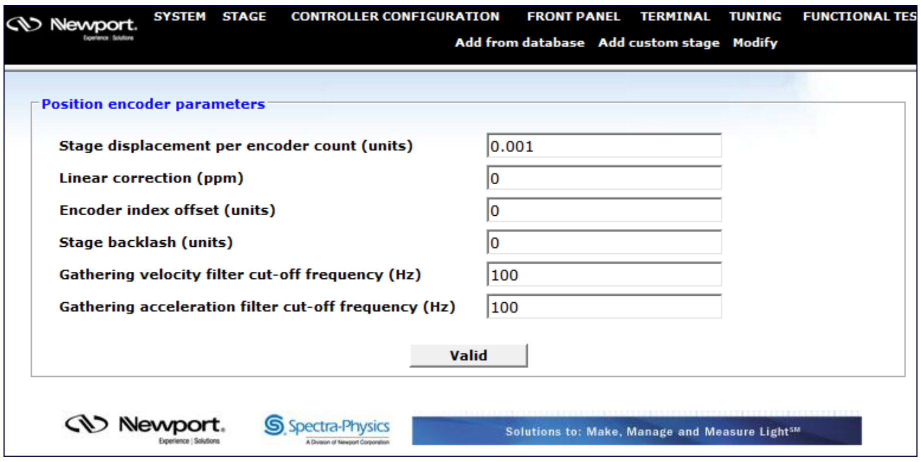


Motor driver parameters

Pulse width modulation frequency (kHz)	50
Velocity servo loop proportional gain	9
Tachometer gain (per cent)	100

Valid

Figure 3: Example for the motor driver parameters page.
The number and type of parameter depends on the settings for the motor driver model (XPS-DRV01 in this case).

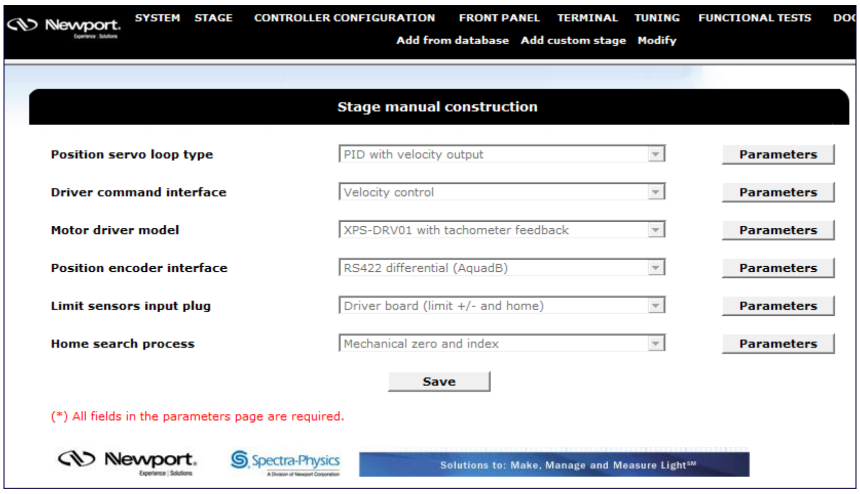


Position encoder parameters

Stage displacement per encoder count (units)	0.001
Linear correction (ppm)	0
Encoder index offset (units)	0
Stage backlash (units)	0
Gathering velocity filter cut-off frequency (Hz)	100
Gathering acceleration filter cut-off frequency (Hz)	100

Figure 4: Example for the position encoder parameters page.
The number and type of parameters depends on the settings for the position encoder interface (RS422 differential (AquadB) in this case).

When all 2nd level settings are done, the *Save* button becomes available (see Figure 5).



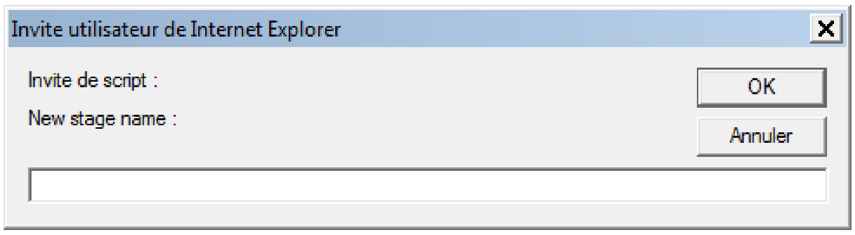
Stage manual construction

Position servo loop type	PID with velocity output
Driver command interface	Velocity control
Motor driver model	XPS-DRV01 with tachometer feedback
Position encoder interface	RS422 differential (AquadB)
Limit sensors input plug	Driver board (limit +/- and home)
Home search process	Mechanical zero and index

(*) All fields in the parameters page are required.

Figure 5: When all 2nd level settings are done, the "Save" button becomes available.

When the "Save" button is pressed, a new dialog box appears (see Figure 6) and prompts the user to specify a stage name.



Invite de script :
New stage name :

Figure 6: Stage name is now required.

Pressing the "OK" button saves the new stage configuration into the "stages.ini" file under the name "UserStageName".

NOTE

During completion, do NOT click on any button as there is the risk of losing all information in the current configuration.

3.0 Position Servo Loop Type

The XPS controller supports 5 different position servo loops:

- PID with a velocity output (primarily used for DC motors with tachometer)
- PID with a motor voltage output (primarily used for DC motors without tachometer)
- PID with an acceleration output (primarily used for linear/brushless and torque motors)
- PI with a position output (primarily used for stepper motors with encoder)
- No servo loop with a position output (primarily used for stepper motors without encoder or to control other motion devices like piezoelectric stages or galvanometric scanners via external amplifiers that feature an analog position input)

All position servo loops have the same structure (cf. Figure 7). The position servo loop compares the SetpointPosition (defined by the profile generator and the group mapping) with the CurrentPosition (reported by the positioner's encoder), to determine the following error. The position servo loop then outputs a value that the controller uses to maintain, increase or decrease the output applied to the driver. The adjustment of the position servo loop parameters allows the user to optimize the performance of their system by increasing or decreasing the responsiveness of the feedback loop.

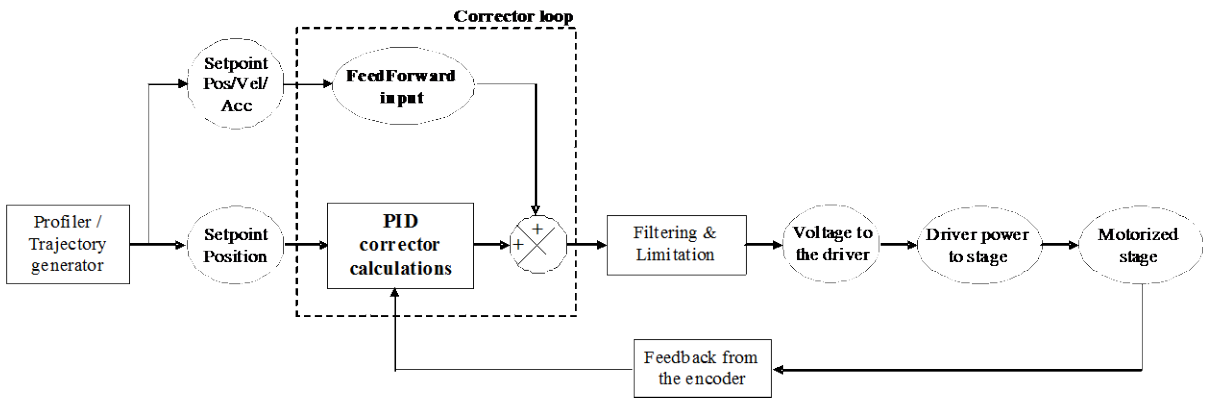


Figure 7: General schematic of a positioner servo loop

3.1 PID with a Velocity Output

Stages.ini file entry: CorrectorType = PIDFFVelocity

This servo loop type is used when a constant value applied to the driver results in constant velocity of the stage, for instance a DC motor with tachometer connected to a driver with internal speed loop.

This servo loop type features a parallel PID servo loop with a feed forward velocity and two notch filters.

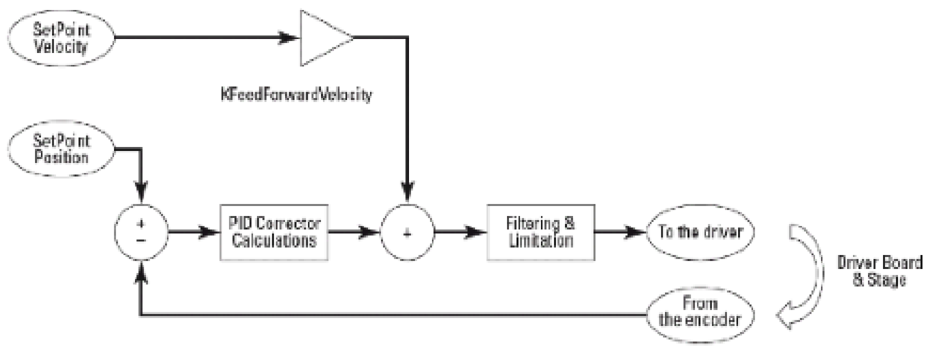


Figure 8: PIDFFVelocity corrector.

3.1.1 Feed Forward and PID Parameters

Needed entries in the configuration file:

- PID servo loop proportional gain: *KP*
- PID servo loop integral gain: *KI*
- PID servo loop derivative gain: *KD*
- PID integral saturation value: *KS*
- PID integration time: *IntegrationTime*
- PID derivative filter cut off frequency: *DerivativeCutOffFrequency*
- velocity feedforward gain: *KFeedForwardVelocity*

The PID servo loop parameters *KP*, *KI*, *KD* and *KFeedForwardVelocity* define the bandwidth of the servo loop. They must be greater than or equal to zero.

The PID integral saturation value *KS* sets the limit of the integral part of the PID servo loop that is applied to the total servo loop output. The value for *KS* must be between 0 and 1.

The PID *IntegrationTime* (seconds) defines the time span for integration of the residual errors. A small value limits the effect of the integral gain *KI*. The value in seconds must be greater than or equal to the servo loop period (125 µs).

The PID *DerivativeFilterCutOffFrequency* (Hz) sets the cut-off frequency of the derivative filter. It can be used to reduce the noise introduced by the numerical derivation of the following error. It must be greater than or equal to zero (zero means filter is disabled) and less than or equal to half of the servo loop frequency (8 kHz).

All parameters will have an impact on system performance and should be set together. It should be noted that parameters are best set for desired performance according the requirements of the motion application (small following error during motion, short settling time after a displacement…). This document will present context specific information on setting PID parameters, but is not intended to be a tutorial on servo loops. Please refer to the common literature for a general treatment of servo loops.

Choice: PI servo loop (Kd = 0)

$C(p) = KP + \frac{KI}{p}$, where *p* is the Laplace variable.

Mechanical transfer function: $H(p) = \frac{1}{p \times (\tau_v \times p + 1)}$

where τ_v is the time constant of the velocity transfer function (s) (cf. § 5.1.2)

Assumption

The position closed loop resonance time constant τ_p is much greater than τ_v. This allows disregarding the velocity servo loop transfer function. By experience, τ_p is about 10-20 ms for most screw driven stages and τ_p / τ_v ranges between 10 and 20.

The damping factor of the closed loop ζ is set to $\zeta = 1.25$ to avoid overshoots, see closed transfer function numerator.

Notice: $\tau = \frac{1}{2 \times \pi \times f}$

PID Parameters

By taken into account that $\tau_p >> \tau_v$, the closed loop transfer function is:

$$T(p) \approx \frac{\frac{KP}{KI}\times p+1}{\frac{1}{KI}\times p^2+\frac{KP}{KI}\times p+1} \approx \frac{2\times\zeta\times\tau_p\times p+1}{\tau_p^2\times p^2+2\times\zeta\times\tau_p\times p+1}$$

This results in:

- Closed loop cut off frequency: $Fc=\frac{\sqrt{1+2\times\zeta^2+\sqrt{\left(1+2\times\zeta^2\right)^2+1}}}{2\times\pi\times\tau_p}$
- PID servo loop proportional gain: $KP=\frac{2\times\zeta}{\tau_p}=156.25$ with $\tau_p=16ms$
- PID servo loop integral gain: $KI=\frac{1}{\tau_p^2}=3906.25$ with $\tau_p=16ms$
- PID servo loop derivative gain: $KD=0$
- PID integral saturation value: default $KS=0.8$
- PID integration time: $\text{IntegrationTime}=1e^{99}s$ (full integration)
- PID derivative filter cut off frequency: $\text{DerivativeCutOffFrequency}=0$ (disabled)
- velocity feedforward gain: $\text{KFeedForwardVelocity}=1$
- Variable PID parameters

3.1.2 Variable PID Parameters

Needed entries in the configuration file:

- Variable PID proportional gain multiplier: *GKP GKP*
- Variable PID integral gain multiplier: *GKI*
- Variable PID derivative gain multiplier: *GKD*
- Variable PID form coefficient: *Kform*

In addition to the classical gains of the PID servo loop, the XPS controller PID position servo loop features variable gain factors *GKP*, *GKD*, and *GKI*. These gains can be used to reduce settling times of systems exhibiting non-uniform mechanical behavior or to tighten the servo loop during the final segment of a move. For example, a stage with a high level of friction will have a response dependent on the size of the move: friction is negligible for a large move, but becomes a predominant factor for small moves. For this reason, the required response of the system to reach the commanded position is not the same for small and large moves. The optimum values of PID parameters for small moves are often higher than the optimum values for large moves. Users that do not want to set individual PID gains for different size motions can benefit from the variable use of PID gains. Variable gains are driven by the distance between the target position and the current position. They must be greater than –1.

The parameter PID form coefficient value *Kform* (units) defines the relationship between the distance to the target and the change of the PID gains:

$$Kx=\left(1+GKx\times\frac{Kform}{\left|\text{TargetPosition}-CurrentPosition\right|+Kform}\right)\times Kx$$

It must be greater than or equal to zero.

The smaller the variable PID form coefficient value the sharper is the change of the PID gains.

GKp = 10 Kp = 2
Target Position = 0
Encoder Position = -100 to 100

$$Kp_{(\text{Kform, Encoder Position})} = \left[1 + GK \cdot \left(\frac{\text{Kform}}{|\text{Target Position} - \text{Encoder Position}| + \text{Kform}}\right)\right] \cdot Kp$$

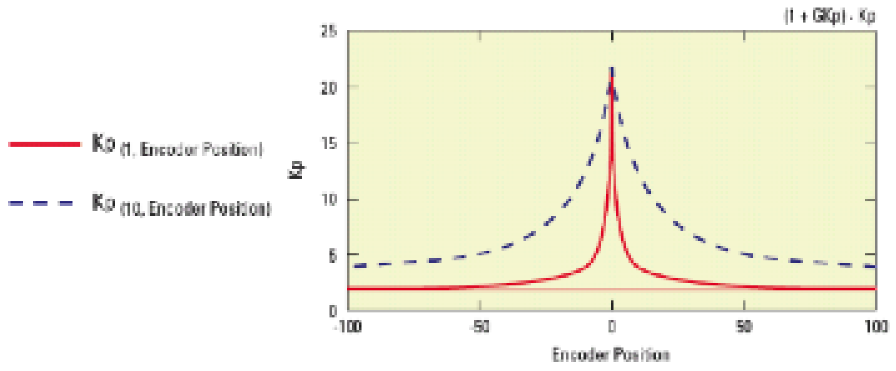


Figure 9: Influence of variable gains.

The default value for these parameters is 0, which disables variable gains.

3.1.3 Position Servo Loop Status

Stages.ini file entry: *ClosedLoopStatus*

The position servo loop status parameter sets the position servo loop either to open loop i.e. without feedback from a position encoder, or to closed loop, i.e. with feedback from a position encoder.

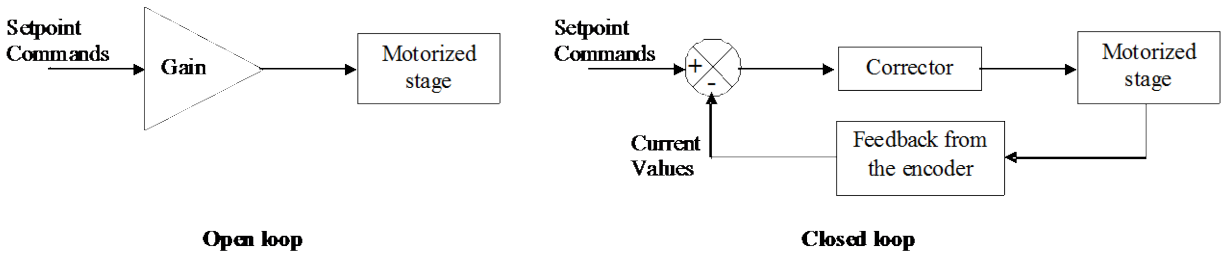


Figure 10: Open and closed loop.

The default value for this parameter is *closed*.

3.1.4 Fatal Following Error

Stages.ini file entry: *FatalFollowingError*

The value for the fatal following error sets the maximum allowed following error of the positioner before generating an error response from the controller. This error is defined as the absolute value of the difference between the setpoint position and the current position. This value is calculated each servo cycle. A following error that exceeds this value will generate the corresponding error code and action. It must be greater than zero.

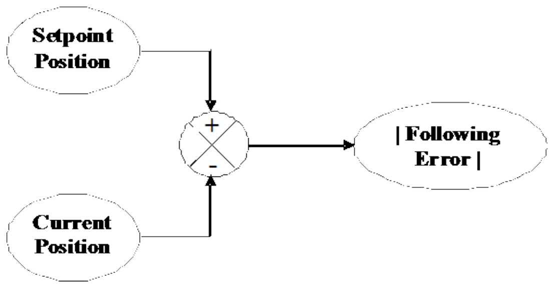


Figure 10: Following error.

3.1.5 Servo Loop Dead Band Threshold

Stages.ini file entry: *DeadBandThreshold*

The servo loop dead band threshold sets the dead band value of the position control loop. When set to a value other than zero, the position loop is disabled when the following error is less than the value for the dead band threshold AND the theoretical motion is done. In some cases, this can avoid oscillations of stages with backlash or friction. It can also reduce stage settling times, but may result in a residual error from the target position. It must be greater than or equal to zero.

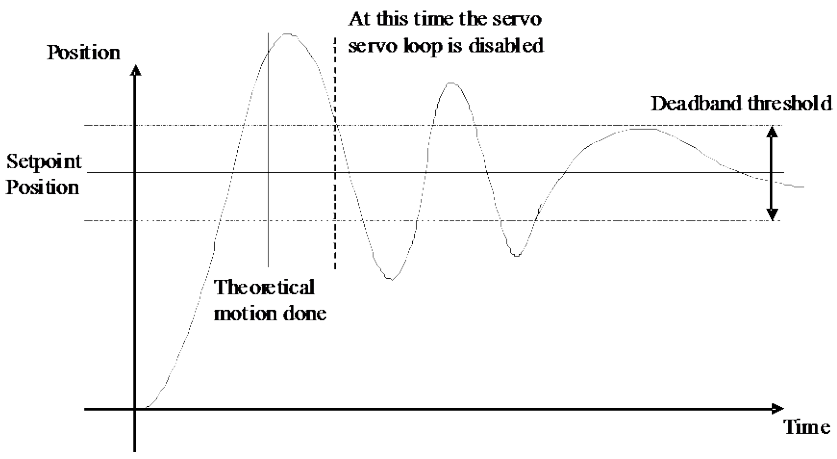


Figure 11: Deadband threshold.

The default value for this parameter is 0, which disables this feature.

3.1.6 Notch Filters Parameters

Needed entries in the configuration file:

- first and second notch filter center frequency: *NotchFrequency1* and *NotchFrequency2*
- first and second notch filter bandwidth: *NotchBandWidth1* and *NotchBandWidth2*
- first and second notch filter gain: *NotchGain1* and *NotchGain2*

The output of the position servo loop is filtered by two notch filters. They can be used to avoid the excitation of specific frequencies. They are defined by their center frequencies *NotchFrequency<n°>* (Hz), bandwidths *NotchBandwidth<n°>* (Hz) and gains *NotchGain<n°>*. The frequencies and bandwidths must be greater than or equal to zero (filter disabled) and less than or equal to half of the servo loop frequency (8 kHz). The gain must be greater than or equal to zero.

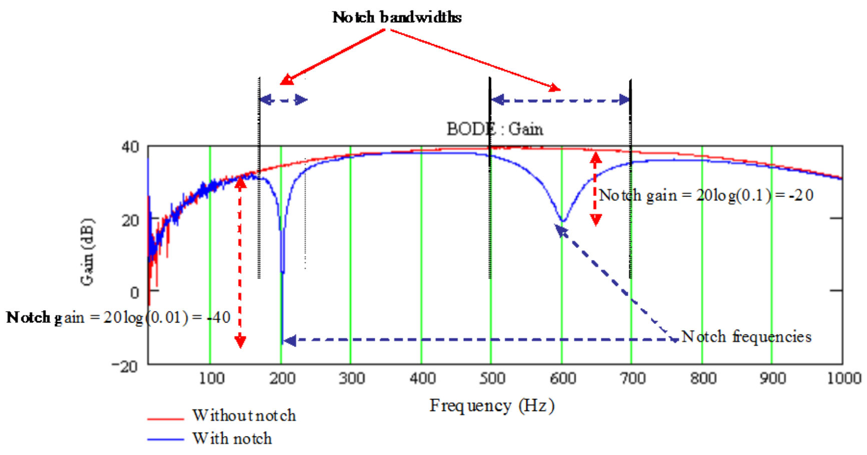


Figure 13: Notch filters.

The default value for these parameters is 0, which disables the filters.

3.1.7 Motion done condition mode

Stages.ini file entry: *MotionDoneMode*

The motion done condition mode defines when a motion is completed. When set to *theoretical motion end,* a motion is completed as defined by the profiler. It does not take into account the settling of the positioner at the end of the move. Therefore, depending on the precision and stability requirements at the end of the move, the theoretical end of a motion might not be always the same as the physical end of a motion. The setting *position and velocity checking* allows a more precise definition of the motion done by conditioning the motion completion to a number of parameters that take the settling of the positioner into account.

For more detailed information about this feature, please refer to the XPS Motion Tutorial, chapter "Motion/Motion Done".

The default value for this parameter is *theoretical motion end*.

3.2 PID with a Motor Voltage Output

Stages.ini file entry: *CorrectorType = PIDDualFFVoltage*

This servo loop type is used when the position servo loop directly drives the voltage applied to the motor, for instance a DC motor without tachometer connected to a voltage amplifier.

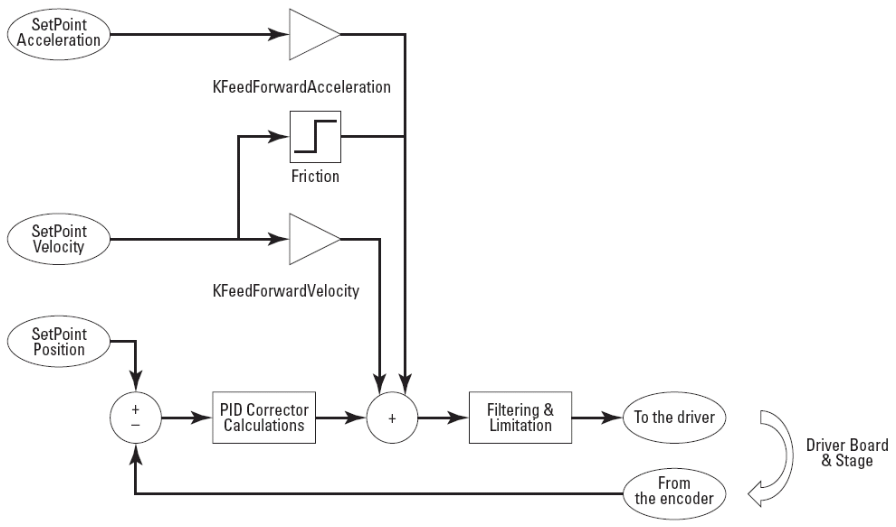


Figure 12: PIDDualFFVoltage corrector.

This servo loop type features a parallel PID servo loop with feed forwards for velocity and acceleration, and two notch filters.

3.2.1 Dual Feed Forward PID Parameters

Needed entries in the configuration file:

- PID servo loop proportional gain: *KP*
- PID servo loop integral gain: *KI*
- PID servo loop derivative gain: *KD*
- PID integral saturation value: *KS*
- PID integration time: *IntegrationTime*
- PID derivative filter cut off frequency: *DerivativeCutOffFrequency*
- Velocity feedforward gain: *KFeedForwardVelocity*
- acceleration feedforward gain: *KFeedForwardAcceleration*
- velocity feedforward gain in open loop: *KFeedForwardVelocityOpenLoop*

The PID servo loop parameters *KP*, *KI*, *KD*, *KFeedForwardVelocity* and *FeedForwardAcceleration* define the bandwidth of the servo loop. They must be greater than or equal to zero.

The PID integral saturation value *KS* sets the limit of the integral part of the PID servo loop that is applied to the total servo loop output. The value for *KS* must be between 0 and 1.

The PID *IntegrationTime* (seconds) defines the time span for integration of the residual errors. A small value limits the effect of the integral gain *KI*. It must be greater than or equal to the servo loop period (125 µs).

The PID *DerivativeFilterCutOffFrequency* (Hz) sets the cut-off frequency of the derivative filter. It can be used to reduce the noise introduced by the numerical derivation of the following error. It must be greater than or equal to zero (zero means filter is disabled) and less than or equal to half of the servo loop frequency (8 kHz).

As the output of the position servo loop is neither velocity nor acceleration, individual feed forwards for the velocity and for the acceleration can be set. The *KFeedForwardVelocityOpenLoop* is only used when the position servo loop is in open loop.

All parameters will have an impact on system performance and should be set together. It should be noted that parameters are best set for desired performance according the requirements of the motion application (small following error during trajectory motion, short settling time after a displacement, …). This document will present context specific information on setting PID parameters, but is not intended to be a tutorial on servo loops. Please refer to the common literature for a general treatment of servo loops.

Choice: PID servo loop

$C(p) = KP + \frac{KI}{p} + \frac{KD \times p}{(\tau_d \times p + 1)} \approx KP + \frac{KI}{p} + KD \times p$, where p is the Laplace variable.

Mechanical transfer function: $H(p) = \frac{1}{p \times 60 \times G \times Kv \times (\tau_m \times p + 1)}$

Where: G: ratio between motor rotation and stage displacement (revolution/units)

Kv: motor voltage constant (V/rpm)

$\tau_m = \frac{R_{mot} \times J}{Kt^2}$: stage mechanical time constant (s)

Where: R_{mot}: motor winding resistance per phase (Ω)

J: total inertia on the motor axis (kg.m²) (cf. § 5.1.2 for detail)

Kt: motor torque constant (N.m/A)

Assumptions

- Position closed loop time constants (real and resonance): $\tau_1 = \tau_2$ (s) is between 8 ms and 16 ms (10 Hz and 20 Hz) depending on the first mechanical resonance.
- The damping factor of the closed loop to avoid overshoots (see closed transfer function numerator) is: $\zeta = 1.25$

Notice: $\tau = \frac{1}{2 \times \pi \times f}$

PID parameters:

The closed loop transfer function is:

$$T(p) \approx \frac{\frac{KD}{KI} \times p^2 + \frac{KP}{KI} \times p + 1}{\frac{60 \times G \times Kv \times \tau_m}{KI} \times p^3 + \left(\frac{KD}{KI} + \frac{60 \times G \times Kv}{KI}\right) \times p^2 + \frac{KP}{KI} \times p + 1}$$

$$\approx \frac{\left(2 \times \zeta \times \tau_1 \times \tau_2 + \tau_2^2 - \frac{\tau_1 \times \tau_2^2}{\tau_m}\right) \times p^2 + (\tau_1 + 2 \times \zeta \times \tau_2) \times p + 1}{(\tau_1 \times p + 1) \times (\tau_2^2 \times p^2 + 2 \times \zeta \times \tau_2 + 1)}$$

Identification with $(\tau_1 \times p + 1) \times (\tau_2^2 \times p^2 + 2 \times \zeta \times \tau_2 + 1)$ results in:

- PID servo loop proportional gain: $KP = \frac{(\tau_1 + 2 \times \zeta \times \tau_2) \times 60 \times G \times Kv \times \tau_m}{\tau_1 \times \tau_2^2}$
- PID servo loop integral gain: $KI = \frac{60 \times G \times Kv \times \tau_m}{\tau_1 \times \tau_2^2}$
- PID servo loop derivative gain: $KD = \frac{((2 \times \zeta \times \tau_1 \times \tau_2 + \tau_2^2) \times \tau_m - \tau_1 \times \tau_2^2) \times 60 \times G \times Kv}{\tau_1 \times \tau_2^2}$
- PID integral saturation value: default $KS = 0.8$

- PID integration time: $\text{IntegrationTime} = 1e^{99} s$ (full integration)
- PID derivative filter cut off frequency: $\text{DerivativeCutOffFrequency} = 5000\text{Hz}$ (maximum)
- velocity feedforward gain: $\text{KFeedForwardVelocity} = 60 \times G \times Kv$
- acceleration feedforward gain: $\text{KFeedForwardAcceleration} = \dfrac{R_{mot} \times J}{Kt}$
- velocity feedforward gain in open loop: $\text{KFeedForwardVelocityOpenLoop} = 60 \times G \times Kv$

3.2.2 Variable PID parameters

Needed entries in the configuration file:

- variable PID proportional gain multiplier: *GKP*
- variable PID integral gain multiplier: *GKI*
- variable PID derivative gain multiplier: *GKD*
- variable PID form coefficient: *Kform*

In addition to the classical gains of the PID servo loop, the XPS controller PID position servo loop features variable gain factors *GKP*, *GKD*, and *GKI*. These gains can be used to reduce settling times of systems exhibiting non-uniform behavior or to tighten the servo loop during the final segment of a move. For example, a stage with a high level of friction will have a response which is dependant on the size of the move: friction is negligible for a large move, but becomes a predominant factor for small moves. For this reason, the required response of the system to reach the commanded position is not the same for small and large moves. The optimum values of PID parameters for small moves are often higher than the optimum values for large moves. Users that do not want to set individual PID gains for different size motions can benefit from variable PID gains. Variable gains are driven by the distance between the target position and the current position. They must be greater than -1.

The parameter PID form coefficient value *Kform* (units) defines the relationship between the distance to the target and the change of the PID gains:

$$Kx = \left(1 + GKx \times \frac{Kform}{|\text{TargetPosition} - CurrentPosition| + Kform}\right) \times Kx$$

It must be greater than or equal to zero.

The smaller the variable PID form coefficient value is, the sharper the change of the PID gains.

GKp = 10 Kp = 2
Target Position = 0
Encoder Position = -100 to 100

$$Kp_{(Kform,\ Encoder\ Position)} = \left[1 + GK \cdot \left(\frac{Kform}{|\text{Target Position} - \text{Encoder Position}| + Kform}\right)\right] \cdot Kp$$

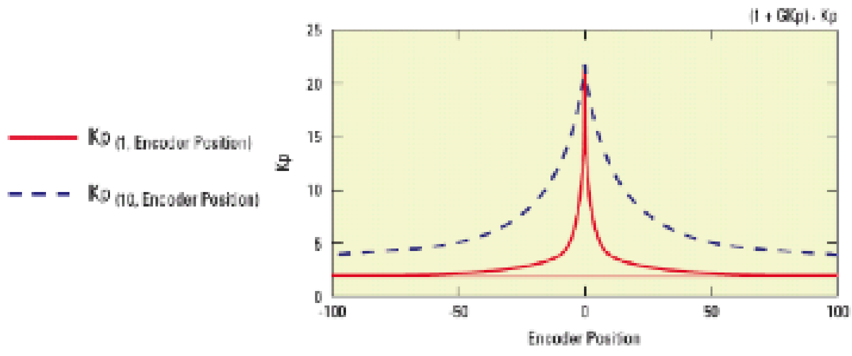


Figure 13: Influence of variable gains.

The default value for these parameters is 0 which disables the variable gains.

3.2.3 Position Servo Loop Status

Stages.ini file entry: *ClosedLoopStatus*

The position servo loop status parameter sets the position servo loop either to open loop i.e. without feedback of a position encoder, or to closed loop, i.e. with feedback from a position encoder.

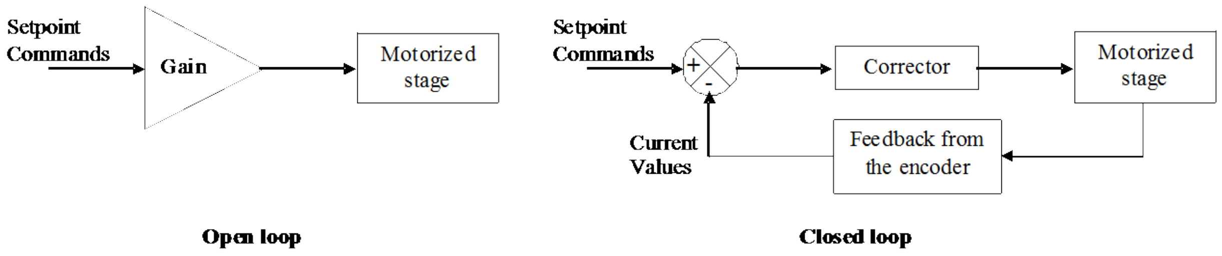


Figure 14: Open and closed loop.

The default value for this parameter is *closed*.

3.2.4 Fatal Following Error

Stages.ini file entry: *FatalFollowingError*

The value for the fatal following error sets the maximum allowed following error of the positioner before generating an error response from the controller. This error is defined as the absolute value of the difference between the setpoint position and the current position. This value is calculated each servo cycle. A following error that exceeds this value will generate the corresponding error code and action. It must be greater than zero.

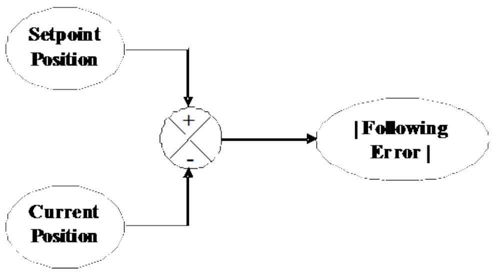


Figure 15: Following error.

3.2.5 Servo Loop Dead Band Threshold

Stages.ini file entry: *DeadBandThreshold*

The servo loop dead band threshold sets the dead band value of the position loop. When set to a value other than zero, the position loop is disabled when the following error is less than the value for the dead band threshold AND the theoretical motion is done. In some cases, this can avoid oscillations of stages with backlash or friction. It can also reduce stage settling times, but may result in residual error from the target position. It must be greater than or equal to zero.

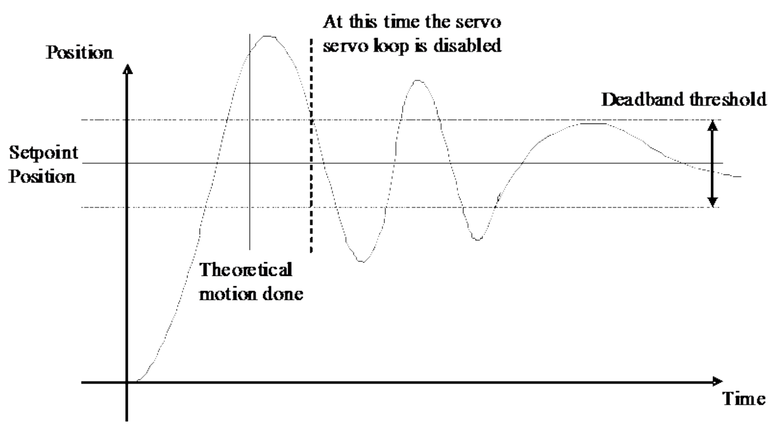


Figure 16: Deadband threshold.

The default value for this parameter is 0, which disables this feature.

3.2.6 Notch Filters Parameters

Needed entries in the configuration file:

- first and second notch filter center frequency: *NotchFrequency1* and *NotchFrequenc y2*
- first and second notch filter bandwidth: *NotchBandWidth1* and *NotchBandWidth2*
- first and second notch filter gain: *NotchGain1* and *NotchGain2*

The output of the position servo loop is filtered by two notch filters. These filters can be used to avoid the excitation of specific frequencies. They are defined by their center frequencies *NotchFrequency<n°>* (Hz), bandwidths *NotchBandwidth<n°>* (Hz) and their gains *NotchGain<n°>*. The frequencies and bandwidths must be greater than or equal to zero (filter disabled) and less than or equal to half of the servo loop frequency (8 kHz). The gain must be greater than or equal to zero.

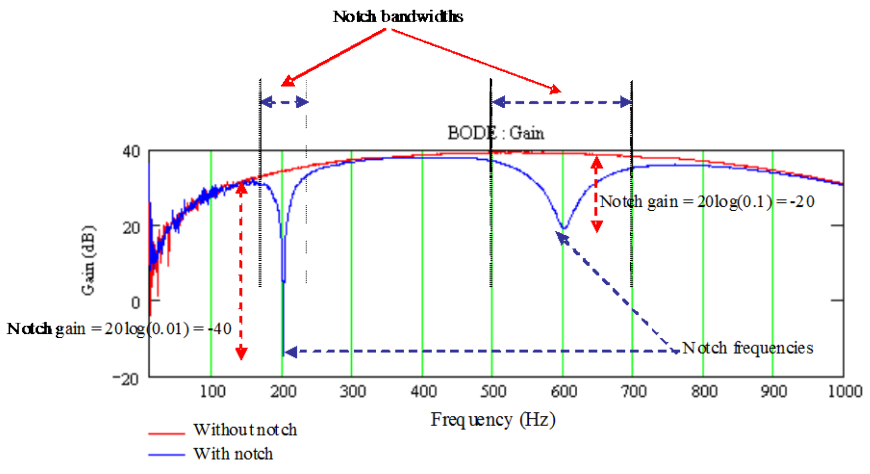


Figure 17: Notch filters.

The default value for these parameters is 0, which disables the filters.

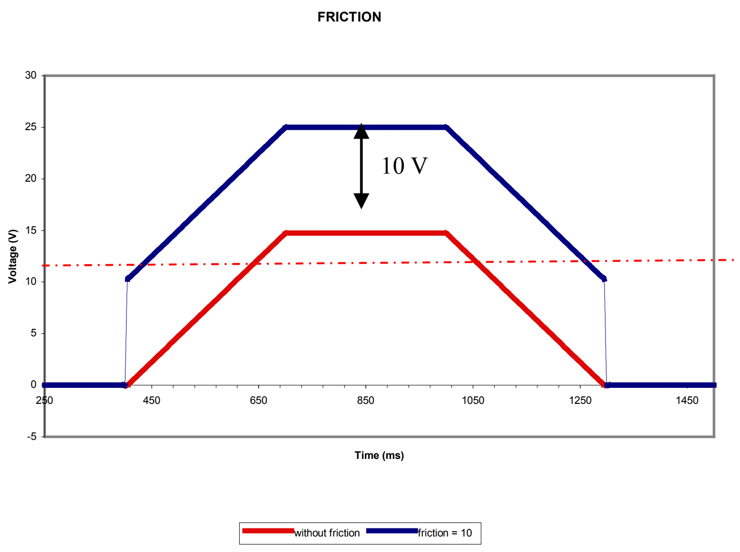
3.2.7 Friction Compensation

Stages.ini file entry: *Friction*

The friction compensation can be used to compensate for friction of stages during motion. When set to a value other than zero, the friction parameter defines a voltage applied directly to the motor consistent with the direction of motion.

The default value for this parameter is 0, which disables this feature.

Displacement of -50 mm



3.2.8 Motion Done Condition Mode

Stages.ini file entry: *MotionDoneMode*

The motion done condition mode defines when a motion is completed. When set to theoretical motion end, a motion is completed as defined by the profiler. It does not take into account the settling of the positioner at the end of the move. Therefore, depending on the precision and stability requirements at the end of the move, the theoretical end of a motion might not be always the same as the physical end of a motion. The setting position and *velocity checking* allows a more precise definition of the motion done by conditioning the motion completion to a number of parameters that take the settling of the positioner into account.

For more detailed information about this feature, please refer to the XPS Motion Tutorial, chapter “Motion/Motion Done”.

The default value for this parameter is *theoretical motion end*.

3.3 PID with an Acceleration Output

Stages.ini file entry: *CorrectorType = PIDFFAcceleration*

This servo loop type is used when a constant value applied to the driver results in a constant acceleration of the stage.

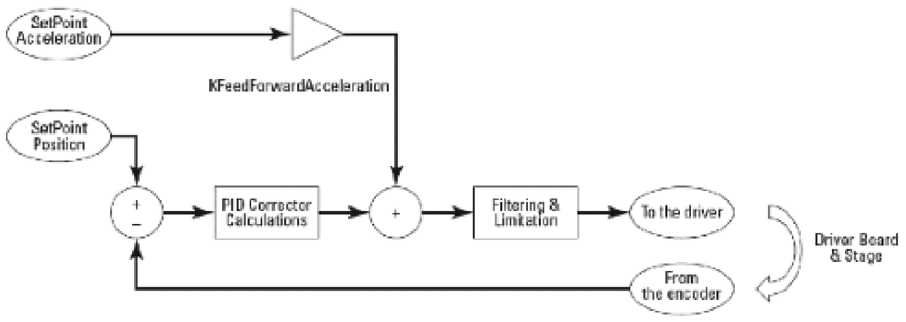


Figure 18: PIDFFAcceleration corrector.

This servo loop type features a parallel PID servo loop with feedforward acceleration and two notch filters.

3.3.1 Feed Forward PID Parameters

Needed entries in the configuration file:

- PID servo loop proportional gain: *KP*
- PID servo loop integral gain: *KI*
- PID servo loop derivative gain: *KD*
- PID integral saturation value: *KS*
- PID integration time: *IntegrationTime*
- PID derivative filter cut off frequency: *DerivativeCutOffFrequency*
- Acceleration feedforward gain: *KFeedForwardAcceleration*
- Jerk feedforward gain: *KFeedForwardJerk*

The PID servo loop parameters *KP*, *KI*, *KD* and *KFeedForwardAcceleration, KFeedForwardJerk* define the bandwidth of the servo loop. They must be greater than or equal to zero.

The PID integral saturation value *KS* sets the limit of the integral part of the PID servo loop that is applied to the total servo loop output. The *KS* parameter must be between 0 and 1.

The PID *IntegrationTime* (seconds) defines the time span for integration of the residual errors. A small value limits the effect of the integral gain *KI*. It must be greater than or equal to the servo loop period (125 µs).

The PID *DerivativeFilterCutOffFrequency* (Hz) sets the cut-off frequency of the derivative filter. It can be used to reduce the noise introduced by the numerical derivation of the following error. It must be greater than or equal to zero (zero means filter is disabled) and less than or equal to half of the servo loop frequency (8 kHz).

All parameters will have an impact on system performance and should be set together. It should be noted that parameters are best set for desired performance according the requirements of the motion application (small following error during trajectory motion, short settling time after a displacement, …). This document will present context specific information on setting PID parameters, but is not intended to be a tutorial on servo loops. Please refer to the common literature for a general treatment of servo loops.

Choice: PID Servo Loop

$C(p) = KP + \frac{KI}{p} + \frac{KD \times p}{(\tau_d \times p + 1)} \approx KP + \frac{KI}{p} + KD \times p$, where *p* is the Laplace variable

Mechanical transfer function: $H(p) = \frac{1}{p^2}$ (driver transfer function disregarded)

Assumptions:

- Position closed loop time constants (real and resonance), $\tau_1 = \tau_2$ (s) is between 4 ms and 8 ms (20 Hz and 40 Hz) depending on the first mechanical resonance.
- The damping factor of the closed loop to avoid overshoots (see closed transfer function numerator) is: $\zeta = 1.25$

Notice: $\tau = \frac{1}{2 \times \pi \times f}$

__PID parameters:__

The closed loop transfer function is:

$$T(p) \approx \frac{\frac{KD}{KI}\times p^2+\frac{KP}{KI}\times p+1}{\frac{1}{KI}\times p^3+\frac{KD}{KI}\times p^2+\frac{KP}{KI}\times p+1} \approx \frac{\left(2\times\zeta\times\tau_1\times\tau_2+\tau_2^2\right)\times p^2+\left(\tau_1+2\times\zeta\times\tau_2\right)\times p+1}{\left(\tau_1\times p+1\right)\times\left(\tau_2^2\times p^2+2\times\zeta\times\tau_2+1\right)}$$

Identification with $\left(\tau_1\times p+1\right)\times\left(\tau_2^2\times p^2+2\times\zeta\times\tau_2+1\right)$ results in:

- PID servo loop proportional gain: $KP=\frac{\tau_1+2\times\zeta\times\tau_2}{\tau_1\times\tau_2^2}=1.4e^5$ with $\tau_1=\tau_2=5ms$
- PID servo loop integral gain: $KI=\frac{1}{\tau_1\times\tau_2^2}=8e^6$ with $\tau_1=\tau_2=5ms$
- PID servo loop derivative gain: $KD=\frac{2\times\zeta\times\tau_1\times\tau_2+\tau_2^2}{\tau_1\times\tau_2^2}=7e^2$ with $\tau_1=\tau_2=5ms$
- PID integral saturation value: default $KS=0.8$
- PID integration time: $\text{IntegrationTime}=1e^{99}s$ (full integration)
- PID derivative filter cut off frequency: $\text{DerivativeCutOffFrequency}=5000\text{Hz}$ (maximum)
- acceleration feedforward gain: $\text{KFeedForwardAcceleration}=1$

3.3.2 Variable PID Parameters

Needed entries in the configuration file:

- variable PID proportional gain multiplier: *GKP*
- variable PID integral gain multiplier: *GKI*
- variable PID derivative gain multiplier: *GKD*
- variable PID form coefficient: *Kform*

In addition to the classical gains of the PID servo loop, the XPS controller PID position servo loop features variable gain factors *GKP*, *GKD*, and *GKI*. These gains can be used to reduce settling times of systems exhibiting non-uniform behavior or to tighten the servo loop during the final segment of a move. For example, a stage with a high level of friction will have a response which is dependant on the size of the move: friction is negligible for a large move, but becomes a predominant factor for small moves. For this reason, the required response of the system to reach the commanded position is not the same for small and large moves. The optimum values of PID parameters for small moves are often higher than the optimum values for large moves. Users that do not want to set individual PID gains for different size motions can benefit from the use of variable PID gains. Variable gains are driven by the distance between the target position and the current position. They must be greater than -1.

The parameter PID form coefficient value *Kform* (units) defines the relationship between the distance to the target and the change of the PID gains:

$$Kx=\left(1+GKx\times\frac{Kform}{\left|\text{TargetPosition}-CurrentPosition\right|+Kform}\right)\times Kx$$

It must be greater than or equal to zero.

The smaller the variable PID form coefficient value is, the sharper is the change of the PID gains.

GKp = 10 Kp = 2

Target Position = 0

Encoder Position = -100 to 100

$$Kp_{(Kform,\ Encoder\ Position)} = \left[1 + GK \cdot \left(\frac{Kform}{|\,Target\ Position - Encoder\ Position\,| + Kform}\right)\right] \cdot Kp$$

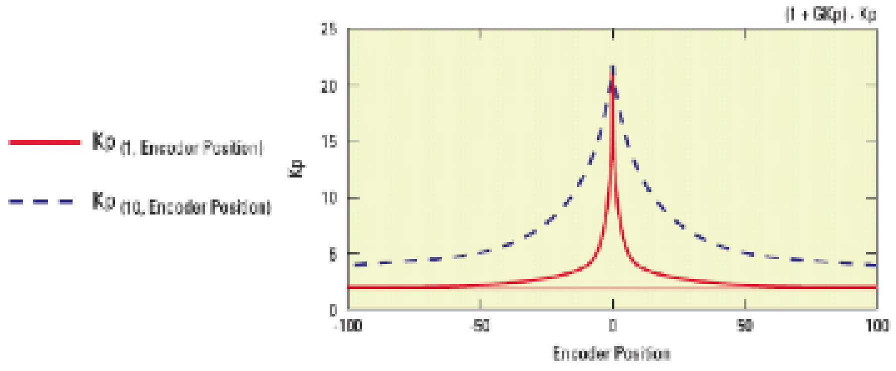


Figure 19: Influence of variable gains.

The default setting for these parameters is 0 which disables the variable gains.

3.3.3 Position servo loop status

Stages.ini file entry: *ClosedLoopStatus*

The position servo loop status parameter sets the position servo loop either to open loop i.e. without feedback of a position encoder, or to closed loop, i.e. with feedback from a position encoder.

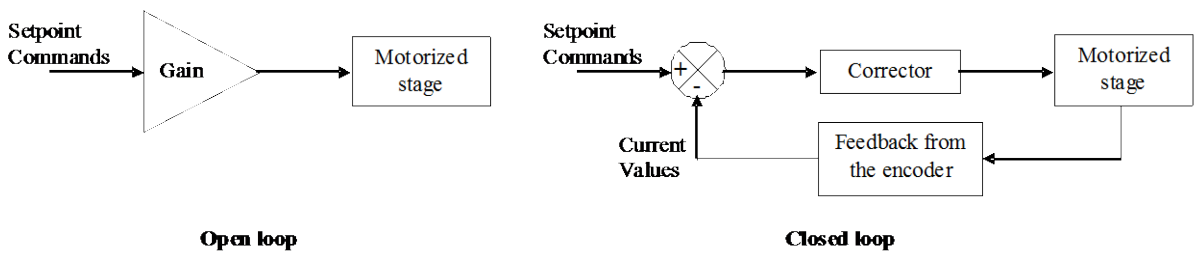


Figure 20: Open and closed loop.

The default value for this parameter is *closed*.

3.3.4 Fatal Following Error

Stages.ini file entry: *FatalFollowingError*

The value for the fatal following error sets the maximum allowed following error of the positioner before generating an error response from the controller. This error is defined as the absolute value of the difference between the setpoint position and the current position. This value is calculated each servo cycle. A following error that exceeds this value will generate the corresponding error code and action. It must be greater than zero.

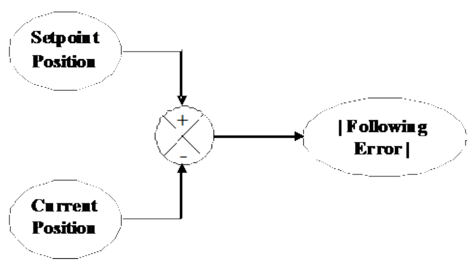


Figure 21: Following error.

3.3.5 Servo Loop Dead Band Threshold

Stages.ini file entry: *DeadBandThreshold*

The servo loop dead band threshold sets the dead band value of the position loop. When set to a value other than zero, the position loop is disabled when the following error is less than the value for the dead band threshold AND the theoretical motion is done. In some cases, this can avoid oscillations of stages with backlash or friction. It can also reduce stage settling times, but may result in residual error from the target position. It must be greater than or equal to zero.

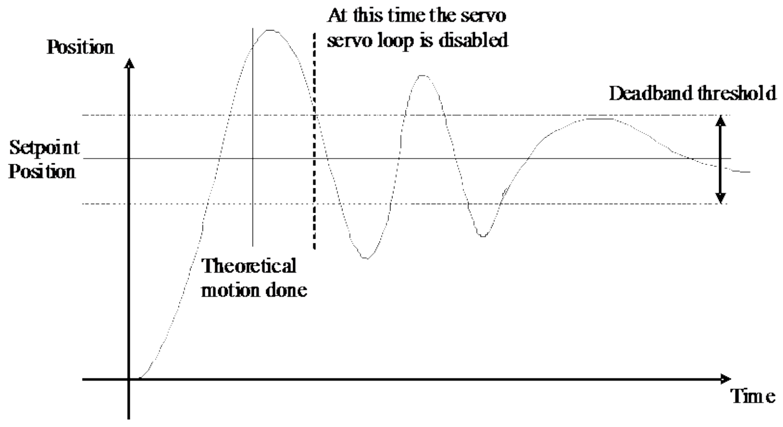


Figure 22: Deadband threshold.

The default value for this parameter is 0, which disables this feature.

3.3.6 Notch filters parameters

Needed entries in the configuration file:

- first and second notch filter center frequency: *NotchFrequency1* and *NotchFrequency2*
- first and second notch filter bandwidth: *NotchBandWidth1* and *NotchBandWidth2*
- first and second notch filter gain: *NotchGain1* and *NotchGain2*

The output of the position servo loop is filtered by two notch filters. These filters can be used to avoid the excitation of specific frequencies. They are defined by their center frequencies *NotchFrequency<n°>* (Hz), bandwidths *NotchBandwidth<n°>* (Hz) and gains *NotchGain<n°>*. The frequencies and bandwidths must be greater than or equal to zero (filter disabled) and less than or equal to half of the servo loop frequency (8 kHz). The gain must be greater than or equal to zero.

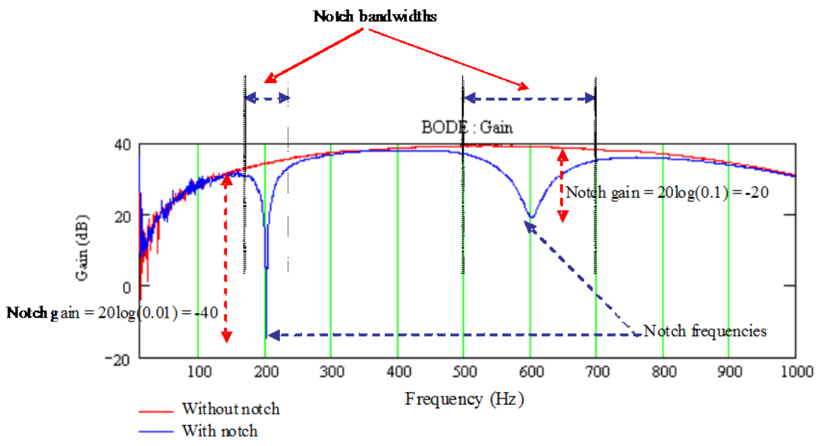


Figure 25: Notch filters.

The default value for these parameters is 0, which disables this feature.

3.3.7 Motion Done Condition Mode

Stages.ini file entry: *MotionDoneMode*

The motion done condition mode defines when a motion is completed. When set to *theoretical motion end,* a motion is completed as defined by the profiler. It does not take into account the settling of the positioner at the end of the move. Therefore, depending on the precision and stability requirements at the end of the move, the theoretical end of a motion might not be always the same as the physical end of a motion. The setting *position and velocity checking* allows a more precise definition of the motion done by conditioning the motion completion to a number of parameters that take the settling of the positioner into account.

For more detailed information about this feature, please refer to the XPS Motion Tutorial, chapter “Motion/Motion Done”.

The default value for this parameter is *theoretical motion end*.

3.4 SR1 (State Return) with an Acceleration Output

Stages.ini file entry: *CorrectorType = SR1Acceleration*

This servo loop type is used when a constant value applied to the driver results in constant acceleration of the stage.

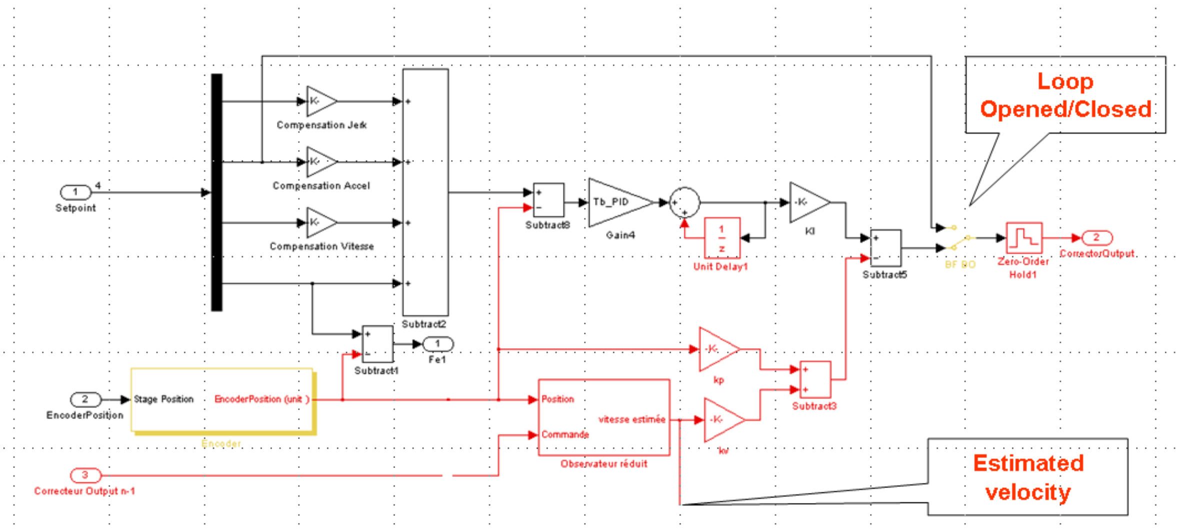


Figure 23: SR1Acceleration corrector.

3.4.1 SR1 parameters

Needed entries in the configuration file:

- SR1 servo loop proportional gain (sec^{-2}): *KP*
- SR1 servo loop integral gain (sec^{-3}): *KI*
- SR1 servo loop velocity gain (sec^{-1}): *KV*
- SR1 observer frequency (Hz): *ObserverFrequency*
- Velocity compensation gain (sec): *CompensationGainVelocity*
- Acceleration compensation gain (sec^2): *CompensationGainAcceleration*
- Jerk compensation gain (sec^3): *CompensationGainJerk*

The SR1 servo loop parameters KP, KI, KV and CompensationGainVelocity, CompensationGainAcceleration, CompensationGainJerk define the bandwidth of the servo loop. KP, KI, KV must be greater than zero, but CompensationGainVelocity, CompensationGainAcceleration and CompensationGainJerk can take any value (zero, positive or negative)

The PID *ObserverFrequency* (Hz) sets the cut-off frequency of the Observer. It is used to reduce the errors introduced by the approximation of the system modeling. It must be greater than or equal to zero (zero means filter is disabled) and less than or equal to half of the servo loop frequency (8 kHz).

All parameters will have an impact on system performance and should be set together. It should be noted that parameters are best set for desired performance according the requirements of the motion application (small following error during trajectory motion, short settling time after a displacement, …).

3.4.2 Position servo loop status

Stages.ini file entry: *ClosedLoopStatus*

The position servo loop status parameter sets the position servo loop either to open loop i.e. without feedback of a position encoder, or to closed loop, i.e. with feedback from a position encoder.

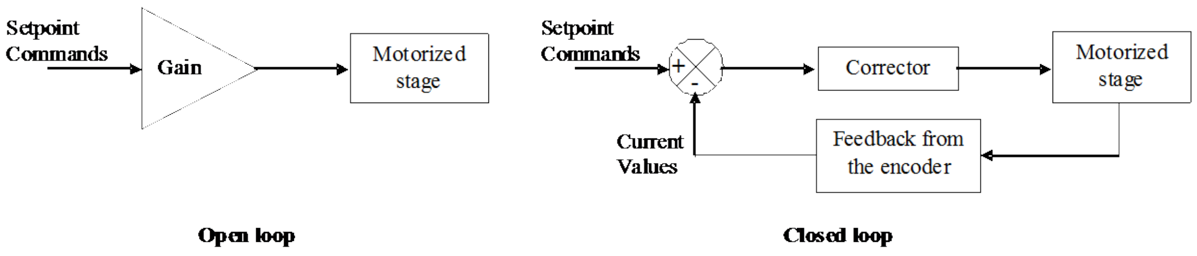


Figure 24: Open and closed loop.

The default value for this parameter is *Closed*.

3.4.3 Fatal following error

Stages.ini file entry: *FatalFollowingError*

The value for the fatal following error sets the maximum allowed following error of the positioner before generating an error response from the controller. This error is defined as the absolute value of the difference between the setpoint position and the current position. This value is calculated each servo cycle. A following error that exceeds this value will generate the corresponding error code and action. It must be greater than zero.

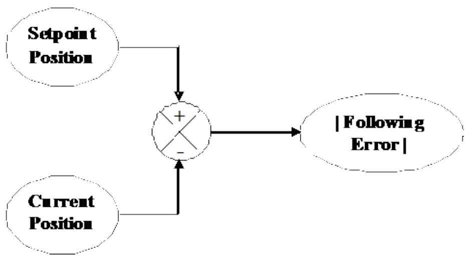


Figure 25: Following error.

3.4.4 Notch Filters Parameters

Needed entries in the configuration file:

- first and second notch filter center frequency: *NotchFrequency1* and *NotchFrequency2*
- first and second notch filter bandwidth: *NotchBandWidth1* and *NotchBandWidth2*
- first and second notch filter gain: *NotchGain1* and *NotchGain2*

The output of the position servo loop is filtered by two notch filters. These filters can be used to avoid the excitation of specific frequencies. They are defined by their center frequencies *NotchFrequency<n°>* (Hz), bandwidths *NotchBandwidth<n°>* (Hz) and gains *NotchGain<n°>*. The frequencies and bandwidths must be greater than or equal to zero (filter disabled) and less than or equal to half of the servo loop frequency (8 kHz). The gain must be greater than or equal to zero.

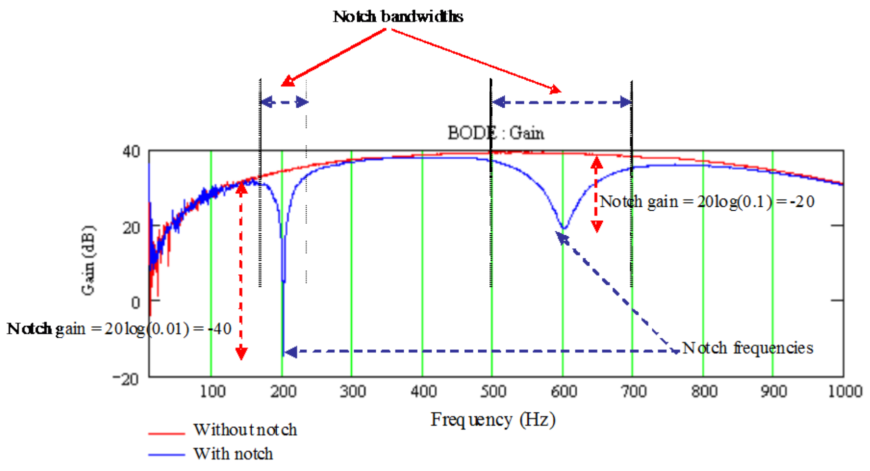


Figure 25: Notch filters.

The default value for these parameters is 0 which disables this feature.

3.4.5 Motion Done Condition Mode

Stages.ini file entry: *CorrectorType = PIPosition*

The motion done condition mode defines when a motion is completed. When set to *theoretical motion end,* a motion is completed as defined by the profiler. It does not take into account the settling of the positioner at the end of the move. Therefore, depending on the precision and stability requirements at the end of the move, the theoretical end of a motion might not be always the same as the physical end of a motion. The setting *position and velocity checking* allows a more precise definition of the motion done by conditioning the motion completion to a number of parameters that take the settling of the positioner into account.

For more detailed information about this feature, please refer to the XPS Motion Tutorial, chapter “Motion/Motion Done”.

The default value for this parameter is *theoretical motion end*.

3.5 PI with a Position Output

Stages.ini file entry: CorrectorType = PIPosition

This servo loop type is used when the position servo loop directly outputs a position value.

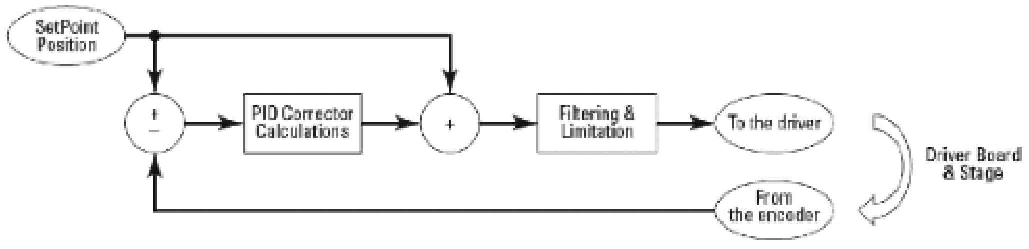


Figure 26: PIPosition corrector.

This servo loop type features a parallel PI servo loop with two notch filters.

3.5.1 PI Parameters

Needed entries in the configuration file:

- PI servo loop proportional gain: *KP*
- PI servo loop integral gain: *KI*
- PI integration time: *IntegrationTime*

The PI servo loop parameters *KP* and *KI* define the bandwidth of the servo loop. They must be greater than or equal to zero.

The PI *IntegrationTime* (seconds) defines the time span for integration of the residual errors. A small value limits the effects of the integral gain *KI*. It must be greater than or equal to the servo loop period (125 µs).

All parameters will have an impact on system performance and should be set together. It should be noted that parameters are best set for desired performance according the requirements of the motion application (small following error during trajectory motion, short settling time after a displacement, …). This document will present context specific information on setting PID parameters, but is not intended to be a tutorial on servo loops. Please refer to the common literature for a general treatment of servo loops.

Choice: I servo loop (no value for *KP*)

$C(p) = \frac{KI}{p}$, where *p* is the Laplace variable.

Mechanical transfer function: $H(p) = 1$

Assumption

The position closed loop real time constant τ_p (s) is between 40 ms and 80 ms (2 Hz to 4Hz)

Notice: $\tau = \frac{1}{2 \times \pi \times f}$

PI parameters

The closed loop transfer function is:

$$T(p) = \frac{1}{\frac{1}{KI} \times p + 1}$$

This results in:

- PI servo loop proportional gain: $KP = 0$
- PI servo loop integral gain: $KI = \frac{1}{\tau_p} = 12.5$ with $\tau_p = 80ms$
- PID integration time: $\text{IntegrationTime} = 1e^{99}s$ (full integration)

3.5.2 Position Servo Loop Status

Stages.ini file entry: *ClosedLoopStatus*

The position servo loop status parameter sets the position servo loop either to open loop i.e. without feedback of a position encoder, or to closed loop, i.e. with feedback from a position encoder.

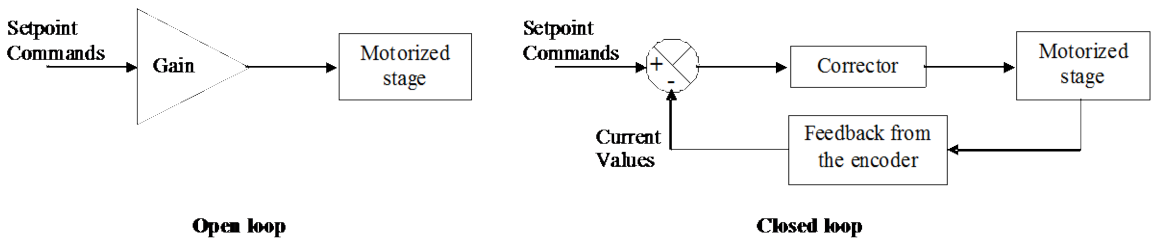


Figure 27: Open and closed loop.

The default value for this parameter is *closed*.

3.5.3 Fatal Following Error

Stages.ini file entry: *FatalFollowingError*

The value for the fatal following error sets the maximum allowed following error of the positioner before generating an error response from the controller. This error is defined as the absolute value of the difference between the setpoint position and the current position. This value is calculated each servo cycle. A following error that exceeds this value will generate the corresponding error code and action. It must be greater than zero.

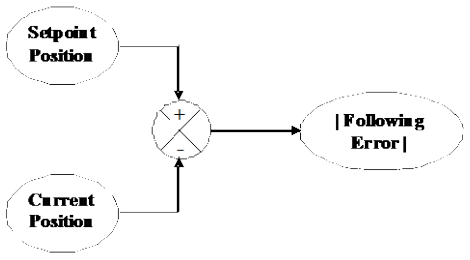


Figure 27: Following error.

3.5.4 Servo Loop Dead Band Threshold

Stages.ini file entry: *DeadBandThreshold*

The servo loop dead band threshold sets the dead band value of the position loop. When set to a value other than zero, the position loop is disabled when the following error is less than the value for the dead band threshold AND the theoretical motion is done. In some cases, this can avoid oscillations of stages with backlash or friction. It can also reduce stage settling times, but may result in residual error from the target position. It must be greater than or equal to zero.

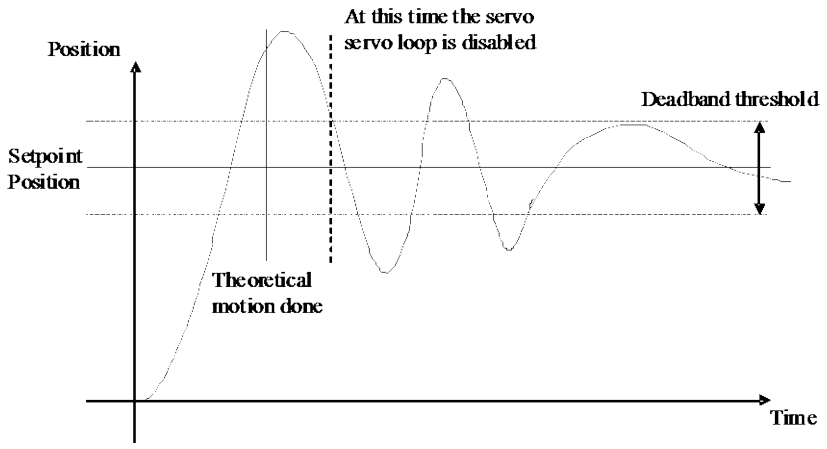


Figure 28: Deadband threshold.

The default value for this parameter is 0 which disables this feature.

3.5.5 Notch Filters Parameters

Needed entries in the configuration file:

- first and second notch filter center frequency: *NotchFrequency1* and *NotchFrequency2*
- first and second notch filter bandwidth: *NotchBandWidth1* and *NotchBandWidth2*
- first and second notch filter gain: *NotchGain1* and *NotchGain2*

The output of the position servo loop is filtered by two notch filters. These filters can be used to avoid the excitation of specific frequencies. They are defined by their center frequencies *NotchFrequency<n°>* (Hz), bandwidths *NotchBandwidth<n°>* (Hz) and gains *NotchGain<n°>*. The frequencies and bandwidths must be greater than or equal to zero (filter disabled) and less than or equal to half of the servo loop frequency (8 kHz). The gain must be greater than or equal to zero.

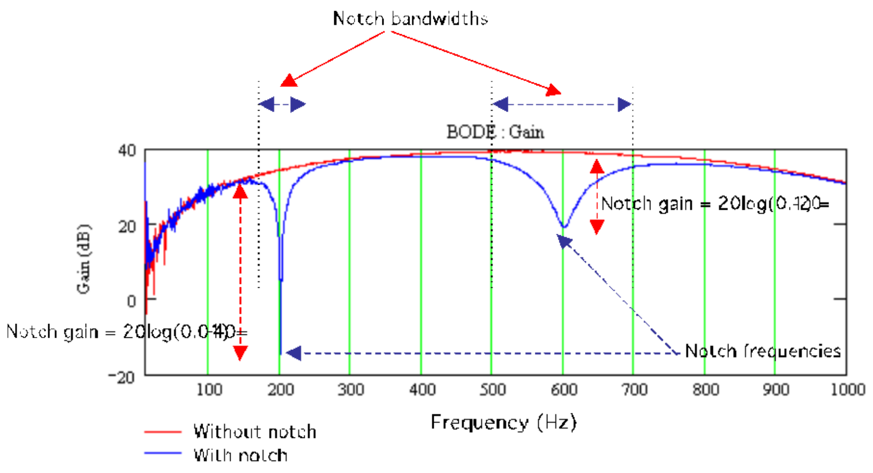


Figure *29*: Notch filters.

The default setting for these parameters is 0 which disables the filters.

3.5.6 Motion Done Condition Mode

Stages.ini file entry: *MotionDoneMode*

The motion done condition mode defines when a motion is completed. When set to *theoretical motion end,* a motion is completed as defined by the profiler. It does not take into account the settling of the positioner at the end of the move. Therefore, depending on the precision and stability requirements at the end of the move, the theoretical end of a motion might not be always the same as the physical end of a motion. The setting *position and velocity checking* allows a more precise definition of the motion done by conditioning the motion completion to a number of parameters that take the settling of the positioner into account.

For more detailed information about this feature, please refer to the XPS Motion Tutorial, chapter “Motion/Motion Done”.

The default value for this parameter is *theoretical motion end.*

3.6 No servo loop with a position output

Stages.ini file entry: *CorrectorType = NoEncoderPosition*

This servo loop type is used for stages without encoder, i.e. stages used always in open loop. There are no parameters to set for this servo loop type.

4.0 Driver Command Interface

The XPS controller features 7 different driver command interfaces:

- velocity control
- voltage control
- acceleration control
- sine/cosine position control
- position control
- 60/90/120 deg UV phase acceleration control
- 60/90/120 deg UV phase dual output acceleration control

The choice of the driver command interface depends on the position servo loop type, the driver type and the motor type.

4.1 Velocity Control

Stages.ini file entry: *MotorInterfaceType = AnalogVelocity*

This driver command interface is used when the output of the position servo loop refers to a velocity value and when the driver input is an analog velocity value. For instance, this is the case with a DC motor with tachometer connected to a driver with internal speed loop and a Position servo loop type setting to *PID with a velocity output*.

This driver command interface also provides a configurable current limitation output.

4.1.1 Stage Velocity at Maximum Command

Stages.ini file entry: *ScalingVelocity*

The stage velocity at maximum command, *ScalingVelocity* (units/s), scales the output of the controller. The value corresponds to the velocity of the positioner with a +10 V input signal to the driver. For the XPS-DRV03 driver board, it is recommended to set this value equal to the maximum allowed stage velocity. For XPS-DRV01 driver board, see § 5.1.2 with the driver board settings.

The value for the *ScalingVelocity* must be greater than zero.

4.1.2 Maximum Allowed Stage Velocity

Stages.ini file entry: *VelocityLimit*

This parameter should not be confused with the *profile generator maximum velocity*, which is the maximum velocity a stage can be commanded to move (cf. § 7.2.4). In order to decrease following errors, a positioner must be capable of moving faster than the *profile generator maximum velocity*. The maximum value is defined by the maximum allowed stage velocity, *VelocityLimit* (units/s).

The higher the dynamic bandwidth of a system, the greater the margin between the *maximum allowed stage velocity* and the *profile generator maximum velocity*.

The value for the *maximum allowed stage velocity* must be less than the *stage velocity at maximum command* and greater than or equal to the *profile generator maximum velocity.* The recommended value is 1.2 times the value for the *profile generator maximum velocity*.

4.1.3 Motor Current at Maximum Command

Stages.ini file entry: *ScalingCurrent*

The motor current at maximum command, *ScalingCurrent* (A), scales the output of the controller for the current limitation setting.

When used with driver board XPS-DRV01, this value must be set to 3 A.

When used with driver board XPS-DRV03, this value must be set to 5 A.

When used with driver board XPS-DRV00, this value scales the 10 V analog output of the output channel B, pin 13 on the XPS-DRV00 (see also § 4.1.4). This value must be greater than zero.

4.1.4 Maximum Allowed Motor Current

Stages.ini file entry: *CurrentLimit*

The maximum allowed motor current, *CurrentLimit* (A), defines the current limitation of the motor driver. This values must be less than or equal to the *motor current at maximum command* and greater than zero.

When used with the driver board XPS-DRV00, this value defines the voltage that gets output on the analog output channel B, pin 13, in relation to the *motor current at maximum command* as follows: Output voltage = 10 V * CurrentLimit (A)/ScalingCurrent (A).

The *CurrentLimit* can be determined as follows:

Motor data

- motor torque constant: Kt (N.m/A)
- maximum allowed motor current: *MotorCurrentLimit* (A)

Driver data:

- motor current at maximum command: *ScalingCurrent* (A)
  (for instance 3A for XPS-DRV01 or 5A for XPS-DRV03)

Stage data:

- ratio between motor rotation and stage displacement: G (revolution/units)
- total inertia on the motor axis: J (kg.m²)

Notice: $J = J_{motor} + J_{load} + J_{mec}$

with J_{motor}: motor rotor inertia (kg.m²)

J_{load}: load inertia (kg.m²)

J_{mec}: bearing, lead screw, … inertia (kg.m²)

User Performance

- maximum stage acceleration (cf. § 7.2.5): $MaximumAcceleration$ (units/s²)

Maximum Allowed Motor Current

- $MaximumCurrent = \frac{MaximumAcceleration \times J \times 2 \times \pi \times G}{Kt}$
- $CurrentLimit = \min(MaximumCurrent \times 1.5, MotorCurrentLimit, ScalingCurrent)$ (A)

NOTE

It is recommended that the CurrentLimit be 1.5 times the motion profiler maximum current to meet the motion requirements of the default stage dynamics.

See also § 5.5.9 for XPS-DRV03 driver board with a RMS limitation.

4.2 Voltage Control

Stages.ini file entry: *MotorInterfaceType = AnalogVoltage*

This driver command interface is used when the output of the position servo loop refers to a motor voltage value and when the driver input is a motor voltage value. For instance, this is the case for a DC motor without tachometer connected to a voltage amplifying driver and a Position servo loop type setting to *PID with motor voltage output*.

This driver command interface also provides a configurable current limitation output.

4.2.1 Motor Voltage at Maximum Command

Stages.ini file entry: *ScalingVoltage*

The motor voltage at maximum command, *ScalingVoltage* (V), scales the analog output of the controller. This value corresponds to the voltage output of the driver with a +10 V input signal. It must be greater than zero.

When used with the driver board XPS-DRV01 or XPS-DRV03, this value must be set to 48 V.

4.2.2 Maximum Allowed Motor Voltage

Stages.ini file entry: *VoltageLimit*

The maximum allowed motor voltage, *VoltageLimit* (V), sets the maximum allowed output voltage of the driver. This value must be less than or equal to the *motor voltage at maximum command* and greater than zero.

This parameter can be determined as follows:

Motor data

- motor winding resistance per phase: R_{mot} (Ω)
- motor torque constant: Kt (N.m/A)
- motor voltage constant: Kv (V/rpm)
- maximum allowed motor current: *MotorCurrentLimit* (A)
- maximum allowed motor voltage: *MotorVoltageLimit* (V)

Driver data

- motor current at maximum command: *ScalingCurrent* (A)
  (for instance 3A for XPS-DRV01 or 5A for XPS-DRV03)
- motor voltage at maximum command; *ScalingVoltage* (V)
  (for instance 48 V for XPS-DRV01 or XPS-DRV03)

Stage data:

- ratio between motor rotation and stage displacement: G (revolution/units)
- total inertia on the motor axis: J (kg.m²)

Notice: $J = J_{motor} + J_{load} + J_{mec}$

with J_{motor}: motor rotor inertia (kg.m²)

J_{load}: load inertia (kg.m²)

J_{mec}: bearing, lead screw, … inertia (kg.m²)

User performance:

- maximum stage velocity (cf. § 7.2.4): $MaximumVelocity$ (units/s)

- maximum stage acceleration (cf. § 7.2.5): $MaximumAcceleration$ (units/s²)

<u>Maximum Allowed Motor Voltage</u>

- $MaximumCurrent = \min\left(\frac{MaximumAcceleration \times J \times 2 \times \pi \times G}{Kt}, MotorCurrentLimit, ScalingCurrent\right)$
- $MaximumVoltage = R_{mot} \times MaximumCurrent + MaximumVelocity \times 60 \times G \times Kv$
- $VoltageLimit = \min(MaximumVoltage \times 1.5, MotorVoltageLimit, ScalingVoltage)$

NOTE

It is recommended that the VoltagaeLimit be 1.5 times the motion profiler maximum voltage to meet the motion requirements of the default stage dynamics.

4.2.3 Motor Current at Maximum Command

Stages.ini file entry: ScalingCurrent

The motor current at maximum command, *ScalingCurrent* (A), scales the output of the controller for the current limitation setting.

When used with the driver board XPS-DRV01, this value must be set to 3 A.

When used with the driver board XPS-DRV03, this value must be set to 5 A.

When used with the driver board XPS-DRV00, this value scales the 10 V analog output of the output channel B, pin 13 on the XPS-DRV00 (see also § 4.1.4). Its value must be greater than zero.

4.2.4 Maximum Allowed Motor Current

Stages.ini file entry: *CurrentLimit*

The maximum allowed motor current, *CurrentLimit* (A), defines the current limit of the motor driver. This value must be less than or equal to the *motor current at maximum command* and greater than zero.

When used with the driver board XPS-DRV00, this value defines the voltage that gets output on the analog output channel B, pin 13, in relation to the *motor current at maximum command* as follows: Output voltage = 10 V * CurrentLimit (A)/ScalingCurrent (A).

The *CurrentLimit* can be determined as follows:

<u>Motor data</u>

- motor torque constant: Kt (N.m/A)
- maximum allowed motor current: *MotorCurrentLimit* (A)

<u>Driver data</u>

- motor current at maximum command: *ScalingCurrent* (A)
  (for instance 3A for XPS-DRV01 or 5A for XPS-DRV03)

<u>Stage data</u>

- ratio between motor rotation and stage displacement: G (revolution/units)
- total inertia on the motor axis: J (kg.m²)

Notice: $J = J_{motor} + J_{load} + J_{mec}$

with J_{motor}: motor rotor inertia (kg.m²)

J_{load}: load inertia (kg.m²)

J_{mec}: bearing, lead screw, … inertia (kg.m²)

User performance

- maximum stage acceleration (cf. § 7.2.5): $MaximumAcceleration$ (units/s²)

Maximum allowed motor current:

- $MaximumCurrent = \frac{MaximumAcceleration \times J \times 2 \times \pi \times G}{Kt}$
- $CurrentLimit = \min(MaximumCurrent \times 1.5, MotorCurrentLimit, ScalingCurrent)$ (A)

NOTE

It is recommended that the CurrentLimit be 1.5 times the motion profiler maximum current to meet the motion requirements of the default stage dynamics.

See also § 5.7.1 for XPS-DRV03 driver board with a RMS limitation.

4.3 Acceleration Control

Stages.ini file entry: *MotorInterfaceType = AnalogAcceleration*

This driver command interface is used when the output of the position servo loop refers to an acceleration value and when the driver input is an analog acceleration value.

4.3.1 Stage Acceleration at Maximum Command

Stages.ini file entry: *ScalingAcceleration*

The stage acceleration at maximum command, *ScalingAcceleration* (units/s²), scales the output of the controller. This value corresponds to the theoretical acceleration reached by the stage with a +10 V input signal to the driver. The value for the *ScalingAcceleration* is stage and driver dependent and must be greater than zero.

See § 5.6.4 for XPS-DRV03 driver board settings.

4.3.2 Maximum allowed stage acceleration

Stages.ini file entry: *AccelerationLimit*

This parameter should not be confused with the *profile generator maximum acceleration*, which is the maximum acceleration a stage can be commanded to move (cf. § 7.2.5). In order to decrease following errors, a positioner must be allowed to move at greater acceleration than the *profile generator maximum acceleration*. Its maximum value is defined by the maximum allowed stage acceleration, *AccelerationLimit* (units/s²). The higher the dynamic bandwidth of a system, the greater the margin between *maximum allowed stage acceleration* and the *profile generator maximum acceleration*.

The value for the *maximum allowed stage acceleration* must be less than the *stage acceleration at maximum command* and greater than or equal to the *profile generator maximum acceleration.* The recommended value is 1.5 times the value of the *profile generator maximum acceleration*.

4.3.3 Stage initialization acceleration level

Stages.ini file entry: *InitializationAccelerationLevel (percent)*

We have the following relation:

InitializationAcceleration = ScalingAcceleration * InitializationAccelerationLevel/100

InitializationAcceleration (units/s²), is the acceleration used during the stage auto-scaling process.

The recommended starting value for this parameter is equal to 20% of the scaling acceleration. If the auto-scaling process does not work properly with this setting (for example, an acceleration that is too low during auto-scaling combined with bad efficiency of the drive chain, could result in failure of the auto-scaling), this value has to be increased step by step. If the displacement during auto-scaling is too large, the *stage initialization acceleration* can be decreased.

4.4 Sine/ Cosine Position Control

Stages.ini file entry: *MotorInterfaceType = AnalogStepperPosition*

This driver command interface is used when the output of the position servo loop is a position value and when the driver inputs are two channels of analog sine/cosine signals. For example, this would be the case for a stepper motor connected to a driver board XPS-DRV01 and a position loop setting to either *PI with position output* or to *No servo loop with position output*.

4.4.1 Motor Current at Maximum Command

Stages.ini file entry: *ScalingCurrent*

The motor current at maximum command, *ScalingCurrent* (A), scales the output of the controller. Its value corresponds to the current output of the driver with a +10 V input signal. It must be greater than zero.

When used with driver board XPS-DRV01, this value must be set to 3 A.

4.4.2 Stage Displacement per Motor Full Step

Stages.ini file entry: *DisplacementPerFullStep*

The stage displacement per motor full step, *DisplacementPerFullStep* (units), defines the stage displacement generated by one full step of the motor. Note: One full step displacement corresponds to ¼ of the electrical period.

The *DisplacementPerFullStep* defines the measurement units of the stage and many parameters are derived from this value, essentially all parameters with units of length, such as velocities or accelerations. Therefore, it is critical this parameter be set correctly for proper operation.

To calculate the *DisplacementPerFullStep* value all of the following must be taken into account: the number of steps per revolution, screw pitch and any gear reduction in the stage.

NOTE

The value for the *maximum velocity* (cf. §7.1.4) must be less than or equal to the *DisplacementPerFullStep* multiplied by the servo loop frequency (8 kHz). This is a requirement for smooth operation, however, we recommend not to exceed one quarter of this maximum possible value.

4.4.3 Stepper Motor Peak Current per Phase

Stages.ini file entry: *PeakCurrentPerPhase*

The stepper motor peak current per phase, *PeakCurrentPerPhase* (A), sets the amplitude of the sine/cosine modulated output current of the driver. This corresponds either to the nominal current per phase (1 phase on) or to the nominal current per phases (2 phases on) multiplied by $\sqrt{2}$. This value can be less than the capability of the motor to reduce motor heating. It must be greater than zero and less than or equal to the *motor current at maximum command.*

4.4.4 Stepper Motor Standby Peak Current per Phase

Stages.ini file entry: *StandbyPeakCurrentPerPhase*

The stepper motor standby peak current per phase, *StandByPeakCurrentPerPhase* (A), sets the amplitude of the sine/cosine modulated output current of the driver when the stage has stopped for 5 s. This parameter allows further reduction of motor heating after a motion. It must be greater than zero and less than or equal to the *stepper motor peak current per phase*.

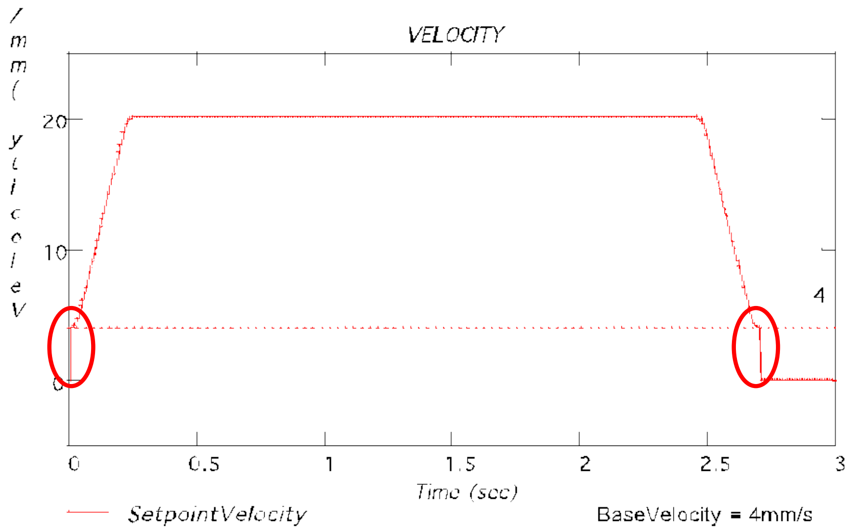
The default value for this parameter is one half of the *stepper motor peak current per phase*.

4.4.5 Stepper Motor Start/Stop Velocity

Stages.ini file entry: *BaseVelocity*

The stepper motor start/stop velocity, *BaseVelocity* (units/s), sets the start/stop velocity of the stepper motor. It must be greater than or equal to zero and less than or equal to the maximum velocity (cf. § 7.2.4).

Example



In this example, the *BaseVelocity* is set to 4 mm/s: during start/stop periods, the velocity passes through the 4mm/s velocity step:

- Start period: Starts motion with 4 mm/s velocity and increases from 4 mm/s to the max velocity according to the settings of the motion profiler
- Stop period: Decreases from the max velocity to 4 mm/s velocity according to the settings of the motion profiler and then decreases to null velocity immediately.

The default value for the *BaseVelocity* is zero.

4.5 Position Control

Stages.ini file entry: *MotorInterfaceType = AnalogPosition*

This driver command interface is used when the output of the position servo loop is a position value and when the driver input is an analog position value. This is the case for some drivers of piezo stages or galvanometric mirrors or voice coils.

4.5.1 Command Voltage at Minimum Target Position

Stages.ini file entry: *MinimumTargetPositionVoltage*

The command voltage at minimum target position, *MinimumTargetPositionVoltage* (V), sets the analog output voltage of the controller when the stage is at its minimum travel limit. This value must be between ±10 V and less than the maximum target position voltage.

4.5.2 Command Voltage At Maximum Target Position

Stages.ini file entry: *MaximumTargetPositionVoltage*

The command voltage at maximum target position, *MaximumTargetPositionVoltage* (V), sets the analog output voltage of the controller when the stage is at its maximum travel limit. This value must be between ±10 V and greater than the minimum target position voltage.

4.6 60/90/120 Deg UV Phase Acceleration Control

Needed entries in the configuration file: *MotorInterfaceType* =

- *AnalogSin60Acceleration*
- *AnalogSin90Acceleration*
- *AnalogSin120Acceleration*

These driver command interfaces are used when the output of the position servo loop refers to a modulated acceleration value and when the driver inputs are analog modulated signals. The modulation is based on the position and suitable for synchronous linear or rotary brushless motors.

Depending on required phase difference between the two output channels, three interfaces are available:

- 60 ° between phases
- 90 ° between phases for two phase motors
- 120 ° between phases for three phase motors

Notice: These interfaces do not require Hall sensors for the motor phase initialization. This is done by a Newport patented process that determines the coil positions relative to the magnetic track solely based on the encoder feedback, and avoids major motion during initialization.

4.6.1 Stage Acceleration At Maximum Command

Stages.ini file entry: *ScalingAcceleration*

The stage acceleration at maximum command, *ScalingAcceleration* (units/s^2), scales the analog output of the controller. This value corresponds to the theoretical acceleration reached by the stage with the maximum voltage input to the driver (+10 V for the amplitude of the sine signal). The *ScalingAcceleration* must be greater than zero and is stage and driver dependent. See also § 5.4.4 for parameter definition with the XPS driver board XPS-DRV02.

4.6.2 Maximum Allowed Stage Acceleration

Stages.ini file entry: *AccelerationLimit*

This parameter should not be confused with the *profile generator maximum acceleration*, which is the maximum acceleration a stage can be commanded to move (cf. § 7.2.5). In order to decrease following errors, a positioner must be allowed to move at greater acceleration than the *profile generator maximum acceleration*. Its maximum value is defined by the maximum allowed stage acceleration, *AccelerationLimit* (units/s^2). The higher the dynamic bandwidth of a system, the greater the margin between *maximum allowed stage acceleration* and the *profile generator maximum acceleration*.

The value for the *maximum allowed stage acceleration* should be set in relation to the application and not solely in relation to the motor capability. For correct parameter determination with the driver board XPS-DRV02, see also § 5.4.4. The value must be less than the *stage acceleration at maximum command* and greater than or equal to the *profile generator maximum acceleration.*

4.6.3 Stage Displacement per Motor Period

Stages.ini file entry: *MagneticTrackPeriod*

The stage displacement per motor period, *MagneticTrackPeriod* (units), is the magnetic pitch between two consecutive north poles of the magnetic track. Its value must be greater than zero.

4.6.4 Stage Initialization Acceleration Level

Stages.ini file entry: *InitializationAccelerationLevel (percent)*

We have the following relation:

*InitializationAcceleration = ScalingAcceleration * InitializationAccelerationLevel/100*

InitializationAcceleration (units/s^2), is the acceleration used during the motor initialization and the stage auto-scaling processes.

The recommended starting value for this parameter is 20% of the scaling acceleration and must be greater than or equal to the maximum acceleration of the profile generator (cf. § 7.2.5). If the initialization or auto-scaling process does not work properly with this setting (for example, an acceleration that is too low during auto-scaling combined with bad efficiency of the drive chain, could result in failure of the initialization or auto-scaling), this value has to be increased step by step. If the displacement during initialization or auto-scaling is too large, the *stage initialization acceleration* can be decreased.

4.6.5 Stage Initialization Cycle Duration

Stages.ini file entry: InitializationCycleDuration

The stage initialization cycle duration, *InitializationCycleDuration* (s), is the time period of the motor initialization process.

This parameter is used only with the LMI (Large Move Initialization) initialization process.

The recommended starting value of this parameter varies from 1 to 20 seconds and is inversely related to the stage friction. For example, a stage with minimal friction requires a longer stage initialization cycle to stabilize the stage during initialization.

4.7 60/90/120 Deg Uv Phase Dual Output Acceleration Control

Needed entries in the configuration file: *MotorInterfaceType =*

- *AnalogDualSin60Acceleration*
- *AnalogDualSin90Acceleration*
- *AnalogDualSin120Acceleration*

These driver command interfaces are used when the output of the position servo loop refers to a modulated acceleration value balanced on two controller axes and when both driver inputs are analog modulated signals. This is a specific interface for driving two motors via two drivers in parallel. The modulation is based on the position and suitable for synchronous linear or rotary brushless motors.

Depending on required phase difference between the two output channels, three interfaces are available:

- 60 ° between phases
- 90 ° between phases for two phase motors
- 120 ° between phases for three phase motors

Notice: These interfaces do not require Hall sensors for the motor phase initialization. This is done by a Newport patented process that determines the coil positions relative to the magnetic track solely based on the encoder feedback, and avoids major motion during initialization.

4.7.1 Stage Acceleration at Maximum Command

Stages.ini file entry: *ScalingAcceleration*

The stage acceleration at maximum command value *ScalingAcceleration* (units/s²) is used to scale the analog output of the controller. This value corresponds to the theoretical acceleration reached by the stage with both motors when the maximum voltage (+10 V for the amplitude of the sine signal) is output by the controller. It must be greater than zero.

See § 5.4.4 for XPS-DRV02 driver board application.

4.7.2 Maximum Allowed Stage Acceleration

Stages.ini file entry: *AccelerationLimit*

The stage acceleration at maximum command, *ScalingAcceleration* (units/s²), scales the analog output of the controller. This value corresponds to the theoretical acceleration reached by the stage with a maximum voltage input to the driver (+10 V for the amplitude of the sine signal). The *ScalingAcceleration* must be greater than zero and is stage and driver dependent. See also § 5.4.4 for parameter definition with the XPS driver board XPS-DRV02.

4.7.3 Stage Displacement per Motor Period

Stages.ini file entry: *MagneticTrackPeriod*

The stage displacement per motor period, *MagneticTrackPeriod* (units), is the magnetic pitch between two consecutive north poles of the magnetic track. This value must be greater than zero.

4.7.4 Stage Initialization Acceleration Level

Stages.ini file entry: InitializationAccelerationLevel (percent)

We have the following relation:

*InitializationAcceleration = ScalingAcceleration * InitializationAccelerationLevel/100*

InitializationAcceleration (units/s²), is the acceleration used during the motor initialization and the stage auto-scaling processes.

The recommended starting value for this parameter is 20% of the scaling acceleration and must be greater than or equal to the maximum acceleration of the profile generator (cf. § 7.2.5). If the initialization or auto-scaling process does not work properly with this setting (for example, an acceleration that is too low during auto-scaling combined with bad efficiency of the drive chain, could result in failure of the initialization or auto-scaling), this value has to be increased step by step. If the displacement during initialization or auto-scaling is too large, the *stage initialization acceleration* can be decreased.

4.7.5 Stage Initialization Cycle Duration

Stages.ini file entry: InitializationCycleDuration

The stage initialization cycle duration, *InitializationCycleDuration* (s), is the time period of the motor initialization process.

This parameter is used only with the LMI (Large Move Initialization) initialization process.

The recommended starting value of this parameter varies from 1 to 20 seconds and is inversely related to the stage friction. For example, a stage with minimal friction requires a longer stage initialization cycle to stabilize the stage during initialization.

4.7.6 Motor Command Input Balance

Needed entries in the configuration file:

- *firstMotorBalance*
- *secondMotorBalance*

The motor command input balance, *FirstMotorBalance* and *SecondMotorBalance*, scale the outputs of the two drivers to apply the acceleration to the center of gravity. The values for both parameters must be between 0 and 1.

4.8 Pulse and Direction Position Control

Required entries in the configuration file: MotorInterfaceType = *PulseDir*

This motor interface drives external stepper motors. The output signals are named PLS_OUT and DIR_OUT. To modify the logic of these signals, two modes are available:

PLS_OUT = pulses generation only

DIR_OUT = direction information only

DigitalStepperDirectionLogic	**Negative**				**Positive**			
DigitalStepperPulseLogic	**Positive**		**Negative**		**Positive**		**Negative**	
DIR_OUT	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**
direction	+	-	+	-	-	+	-	+
PLS_OUT	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**
pulse	□	■	■	□	□	■	■	□

■ = pulse □ = no pulse + = positive direction - = negative direction

4.8.1 Direction Logic

Stages.ini file entry: *DigitalStepperDirectionLogic*

The logic direction can be **Negative** (DIR_OUT: 1 = negative direction and 0 = positive direction) or **Positive** (DIR_OUT: 1 = positive direction and 0 = negative direction).

The PLS_OUT represents the pulse generation.

DIR_OUT	**0**	**1**
DigitalStepperDirectionLogic = **Positive**	Direction -	Direction +
DigitalStepperDirectionLogic = **Negative**	Direction +	Direction -

4.8.2 Pulse Generation Logic

Stages.ini file entry: *DigitalStepperPulseLogic*

The logic pulse can be **Negative** (PLS_OUT: 1 = no pulse and 0 = pulse) or **Positive** (PLS_OUT: 1 = pulse and 0 = no pulse).

PLS_OUT	**0**	**1**
DigitalStepperPulseLogic = **Positive**	No pulse	Pulse
DigitalStepperPulseLogic = **Negative**	Pulse	No pulse

4.8.3 Stage Displacement per Motor Full Step

Stages.ini file entry: *DisplacementPerFullStep*

The stage displacement per motor full step, *DisplacementPerFullStep* (units), defines the stage displacement generated by one full step of the motor.

NOTE

One full step displacement corresponds to ¼ of the electrical period.

The *DisplacementPerFullStep* defines the measurement units of the stage and many parameters are derived from this value, essentially all parameters with units of length, such as velocities or accelerations. It is critical this value be set correctly for proper operation of the stage.

To calculate the *DisplacementPerFullStep* value all of the following must be taken into account: the number of steps per revolution, screw pitch and any gear reduction in the stage.

4.8.4 Number of Micro Steps in the Displacement per Motor Full Step

Stages.ini file entry: *MicroStepsPerFullStep*

The *MicroStepsPerFullStep* defines the number of micro steps in the displacement per one full step of the stepper motor.

4.9 Pulse + and Pulse - position control

Needed entries in the configuration file: *MotorInterfaceType = PulsePulse*

This motor interface drives external stepper motors. The output signals are named PLS_OUT and DIR_OUT. The Pulse + and pulse – mode represents ClockWise and CounterClockWise.

PLS_OUT = direction information and pulse generation

DIR_OUT = direction information and pulse generation

DigitalStepperDirectionLogic	**Negative**				**Positive**			
DigitalStepperPulseLogic	**Positive**		**Negative**		**Positive**		**Negative**	
DIR_OUT	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**
direction	-	-	-	-	+	+	+	+
pulse	□	■	■	□	□	■	■	□
PLS_OUT	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**
direction	+	+	+	+	-	-	-	-
pulse	□	■	■	□	□	■	■	□

■ = pulse □ = no pulse + = positive direction - = negative direction

4.9.1 Direction Logic

Stages.ini file entry: *DigitalStepperDirectionLogic*

The logic direction can be **Negative** (DIR_OUT = negative direction and PLS_OUT = positive direction) or **Positive** (DIR_OUT = positive direction and PLS_OUT = negative direction)

	DIR_OUT	**PLS_OUT**
DigitalStepperDirectionLogic = **Positive**	Direction +	Direction -
DigitalStepperDirectionLogic = **Negative**	Direction -	Direction +

4.9.2 Pulse Generation Logic

Stages.ini file entry: *DigitalStepperPulseLogic*

The logic pulse can be **Negative** (DIR_OUT or PLS_OUT: 1 = no pulse and 0 = pulse) or **Positive** (DIR_OUT or PLS_OUT: 1 = pulse and 0 = no pulse).

PLS_OUT or **DIR_OUT**	**0**	**1**
DigitalStepperPulseLogic = **Positive**	No pulse	Pulse
DigitalStepperPulseLogic = **Negative**	Pulse	No pulse

4.9.3 Stage Displacement per Motor Full Step

Stages.ini file entry: *DisplacementPerFullStep*

The stage displacement per motor full step, *DisplacementPerFullStep* (units), defines the stage displacement generated by one full step of the motor.

NOTE

One full step displacement corresponds to ¼ of the electrical period.

The *DisplacementPerFullStep* defines the measurement units of the stage and many parameters are derived from this value, essentially all parameters with units of length, such as velocities or accelerations. It is critical this value be set correctly for proper operation of the stage.

To calculate the *DisplacementPerFullStep* all of the following must be taken into account: the number of steps per revolution, screw pitch and any gear reduction in the stage.

4.9.4 Number of Micro Steps in the Displacement per Motor Full Step

Stages.ini file entry: *MicroStepsPerFullStep*

The *MicroStepsPerFullStep* defines the number of micro steps in the displacement per one full step of the stepper motor.

4.10 Piezo Position Control

Stages.ini file entry: *MotorInterfaceType = AnalogPositionPiezo*

This type of driver command interface is used specially for piezo stages (DRVP1, …). It is compatible only with *NoEncoderPosition* or *PIPosition* corrector type.

5.0 Motor Driver Model

The XPS controller supports the following settings for the motor driver model:

- XPS-DRV01 with tachometer feedback
- XPS-DRV01 without tachometer feedback
- XPS-DRV01 for stepper motors
- XPS-DRV02 for linear/brushless motors
- XPS-DRV03 with tachometer feedback
- XPS-DRV03 for acceleration control
- XPS-DRV03 for voltage control
- XPS-DRV00 for non-configurable external drivers
- XPS-DRV00P for configurable external drivers
- NON_CONFIGURABLE_DRV for non-configurable external drivers (CIE and external driver are directly connected, without use of DRV00/DRV00P pass-through cards).
- XPS-DRVPx (x = 1, 2, …) for piezo stage drivers.

The choice of a driver board setting depends on the driver board used, the driver command interface, position servo loop type, and motor type.

5.1 XPS-DRV01 with Tachometer Feedback

Stages.ini file entry: *XPS-DRV01*

This type of motor driver model is used for DC motors with tachometer. The driver command interface must be set to *velocity control* (cf. § 4.1) and the position servo loop type to *PID with velocity output (*cf. § 3.1). The XPS-DRV01 driver board supplies a maximum output of 3 A and 48 V.

5.1.1 Pulse width Modulation Frequency

Stages.ini file entry: *DriverName = DriverPWMFrequency*

The pulse width modulation frequency, *DriverPWMFrequency* (Hz), sets the frequency of the pulse width modulation output of the XPS-DRV01 driver.

The default value is 50 kHz.

5.1.2 Velocity Servo Loop Parameters

Required entries in the configuration file:

- velocity servo loop proportional gain: *DriverErrorAmplifierGain*
- tachometer gain: *DriverTachometerGain*

The velocity servo loop proportional gain and the tachometer gain should be set together to optimize the bandwidth of the velocity loop.

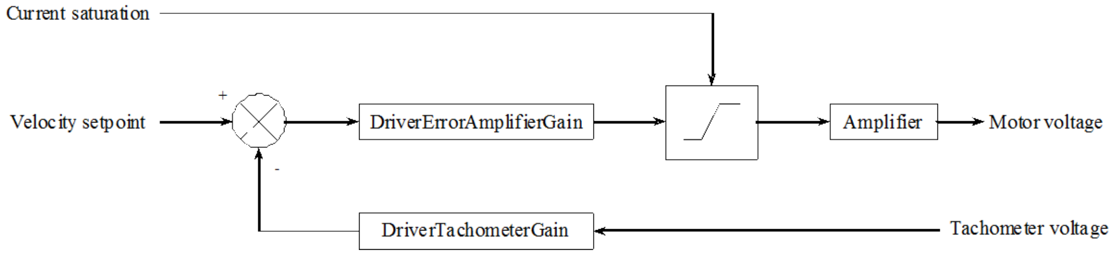


Figure 30: Velocity servo loop.

These parameters are set up as follow.

Motor data

- motor maximum allowed velocity: $V_{motor\ max}$ (rpm)
- motor winding resistance per phase: R_{mot} (Ω)
- motor torque constant: Kt (N.m/A)
- motor voltage constant: Kv (V/rpm)

Notice: $Kt = Kv$ when there are given in these units

Tachometer data

- tachometer voltage constant: K_{GT} (V/rpm)

Driver data

- DC power supply voltage: $U_{DC} = 48V$
- DAC maximum voltage: $U_{DAC\ max} = 10V$
- maximum allowed current at maximum command: $ScalingCurrent = 3A$
- tachometer feedback maximum voltage: $U_{tachometer\ max} = 10V$
- tachometer gain (*DriverTachometerGain*): $K_{tachometer} \in \{100,75,60,50,33,30,27,25,0\}$
- velocity servo loop proportional gain (*DriverErrorAmplifierGain*): $Kp_{velocity} \in \{1,5,9,13,17,21,25,29\}$

Stage data

$\tau_m = \frac{R_{mot} \times J}{Kt^2}$

- ratio between motor rotation and stage displacement: G (revolution/units)
- total inertia on the motor axis: J (kg.m²)

Notice: $J = J_{motor} + J_{load} + J_{mec}$

with J_{motor}: motor rotor inertia (kg.m²)

J_{load}: load inertia (kg.m²)

J_{mec}: bearing, lead screw, … inertia (kg.m²)

- stage mechanical time constant: τ_m (s)

Notice:

User performance

- maximum allowed stage velocity (cf. § 4.1.2): $VelocityLimit$ (units/s)
- maximum stage acceleration (cf. § 7.2.5): $MaximumAcceleration$ (units/s²)
- velocity servo loop cut off frequency: Fc (Hz)

Notice: usually this cut off frequency is between 100Hz and 200 Hz

Preliminary tachometer gain

- maximum motor velocity: $V_{motor\,\mathrm{limit}} = 60 \times G \times V_{\mathrm{limit}} \le V_{motor\,\max}$ (rpm)
- raw tachometer gain: $K_{\mathrm{tachometer}\,raw} = \dfrac{U_{\mathrm{tachometer\,max}}}{V_{motor\,\mathrm{limit}} \times K_{GT}} \times 100$
- tachometer gain: $K_{\mathrm{tachometer}}$ closest lower value

Maximum allowed motor current:

- $CurrentLimit = \dfrac{MaximumAcceleration \times 2 \times \pi \times J \times G}{Kt} \times 1.5 \le ScalingCurrent$ (A)

Notice: the coefficient 1.5 is the margin for parameters uncertainty and servo loop transient.

Velocity servo loop proportional gain

- raw velocity servo loop proportional gain:

$$Kp_{velocit\,raw} = \frac{U_{DAC\,\max} \times Kv \times (\tau_m \times 2 \times \pi \times Fc - 1)}{U_{DC} \times K_{GT} \times \dfrac{K_{\mathrm{tachometer}}}{100}}$$

- velocity servo loop proportional gain: $Kp_{velocity}$ closest higher value

Cut off frequency recalculation due to quantification

- raw cut off frequency: $Fc_{raw} = \dfrac{\dfrac{Kp_{velocity} \times U_{DC} \times K_{GT}}{U_{DAC\,\max} \times Kv} \times \dfrac{K_{\mathrm{tachometer}}}{100} + 1}{\tau_m \times 2 \times \pi}$ (Hz)
- if the value is too far from the velocity servo loop cut off frequency, the tachometer gain

$K_{\mathrm{tachometer}}$ can be decreased.

Stage velocity at maximum command (cf. § 4.1.1):

- $ScalingVelocity = \dfrac{\dfrac{Kp_{velocity} \times U_{DC}}{60 \times G \times Kv}}{\dfrac{Kp_{velocity} \times U_{DC} \times K_{GT}}{U_{DAC\,\max} \times Kv} \times \dfrac{K_{\mathrm{tachometer}}}{100} + 1} \ge VelocityLimit$ (units/s)

5.2 XPS-DRV01 without Tachometer Feedback

Stages.ini file entry: *DriverName = XPS-DRV01*

This type of motor driver model is used with DC motors without tachometer. The driver command interface must be set to *voltage control* (cf. § 4.2) and the position servo loop type to *PID with voltage output (*cf. § 3.2). The XPS-DRV01 driver board supplies a maximum output of 3 A and 48 V.

5.2.1 Pulse width Modulation Frequency

Stages.ini file entry: *DriverPWMFrequency*

The pulse width modulation frequency, *DriverPWMFrequency* (Hz), sets the frequency of the pulse width modulation output of the XPS-DRV01 driver.

The default value is 50 kHz.

5.3 XPS-DRV01 for Stepper Motors

Stages.ini file entry: *DriverName = XPS-DRV01*

This type of motor driver model is used for stepper motors. The driver command interface must be set to *sine/cosine position control* (cf. § 4.4) and the position servo loop type to either *PI with a position output (*cf. § 3.5) or to *no servo loop with a position output* (cf. § 3.6). The XPS-DRV01 driver board supplies a maximum output of 3 A and 48 V.

5.3.1 Pulse width Modulation Frequency

Stages.ini file entry: *DriverPWMFrequency*

The pulse width modulation frequency, *DriverPWMFrequency* (Hz), sets the frequency of the pulse width modulation output of the XPS-DRV01 driver.

The default value is 50 kHz.

5.3.2 Stepper Motor Winding Connection

Stages.ini file entry: *DriverStepperWinding*

The stepper motor winding connection, *DriverStepperWinding*, specifies the stepper wiring. If the stepper is wired between pin 1-4 and pin 5-8, the parameter must be set to *Full.* If the stepper is wired between pin 1-4 and 11-12 as well as pin 5-10, the parameter must be set to *Half.*

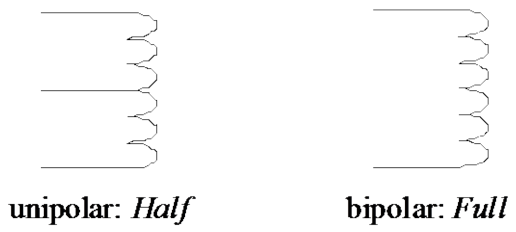


Figure 31: Stepper motor winding connection.

5.4 XPS-DRV02/XPS-DRV02P for Linear/Brushless Motors

Stages.ini file entry:

DriverName = XPS-DRV02 ; for DRV02 driver cards

or

DriverName = XPS-DRV02P ; for DRV02P driver cards

This type of motor driver model is used for brushless linear or rotary motors. The driver command interface must be set to *120 deg UV phase acceleration control* (cf. § 4.6) or *120 deg UV phase dual output acceleration control* (cf. § 4.7) and the position servo loop type to *PID with acceleration output* (cf. §4.6). The XPS-DRV02 driver board supplies a maximum output of 5 A and 44 Vpp. The XPS-DRV02P driver board supplies a maximum output of 7 A and 44 Vpp.

5.4.1 Motor Winding Resistance per Phase

Stages.ini file entry: *DriverMotorResistance*

The motor winding resistance per phase, *DriverMotorResistance* (Ω), is the motor resistance of each phase. It must be greater than zero and less than or equal to 65.535 Ω.

5.4.2 Motor Winding Induction per Phase

Stages.ini file entry: *DriverMotorInductance*

The motor winding induction per phase, *DriverMotorInductance* (H), is the motor induction of each phase. It must be greater than zero and less than or equal to 65.535 mH.

5.4.3 Current Servo Loop Cut Off Frequency

Stages.ini file entry: *DriverCutOffFrequency*

The current servo loop cut off frequency, *DriverCutOffFrequency* (Hz), sets the bandwidth of the internal driver current servo loop. The internal driver PI parameters are calculated automatically. This value has to be set relative to the bandwidth of the position servo loop (cf. § 3.4.1). It must be greater than zero and less than or equal to 3 kHz.

For a stage with a position servo loop cut off frequency between 20 and 50 Hz, the *current servo loop cut off frequency* should be set between 200 and 500 Hz.

5.4.4 Current Monitoring Parameters

Needed entries in the configuration file:

- peak current limit: *DriverMaximumPeakCurrent*
- RMS current limit: *DriverMaximumRMSCurrent*
- RMS integration time: *DriverRMSIntegrationTime*

The peak current limit, *DriverMaximumPeakCurrent* (A), is the maximum allowed motor current. If the motor current goes beyond this value, a driver fault is generated. This value must be greater than zero and less than or equal to 5 A.

The RMS current limit, *DriverMaximumRMSCurrent* (A), is the maximum allowed RMS motor current. The RMS integration time *DriverRMSIntegrationTime* (s) defines the integration time span. If the RMS motor current goes beyond this value, a driver fault is generated. The *RMS Current limit* must be greater than zero and less than or equal to 5 A. The *RMS integration time* must be greater than zero and less than or equal to 480 s.

There are many different methods possible to define these values. Here is a description of one method used by Newport for standard products.

Application input:

- Due to the high accuracy requirements of Newport testing, a maximum temperature increase of the motor coils by 20 °C is set to avoid thermal effects impacting the motion precision: $\Delta T = 20^{o}C$
- A ratio of 2 between the peak current limit and the RMS current limit is used. This has been found to be a realistic approach in numerous precision motion applications as a good balance between throughput and precision: $r = 2$

Motor data

- motor force/torque constant: Kt (N/Arms) or (N.m/Arms)
- motor constant or steepness: S (N²/W) or (N².m²/W)
- thermal resistance: R_{th} (°C/W)
- thermal time constant: τ_{th} (s)

Driver data

- maximum driver current: $I_{\max} = 5A$

Stage data

- load (linear direct drive stage only): *Load* (kg)
- ratio between motor rotation and stage displacement: G (revolution/units)
- total inertia on the motor axis: J (kg.m²)

Notice: $J = J_{motor} + J_{load} + J_{mec}$

with J_{motor}: motor rotor inertia (kg.m²)

J_{load}: load inertia (kg.m²)

J_{mec}: bearing, lead screw, … inertia (kg.m²)

Calculations:

- $Kt = \sqrt{3 \times Rf \times S}$,

where Rf (Ω) is the motor resistance per phase

- $\tau_{th} = \dfrac{R_{th} \times F_p{}^2}{\theta\tau \times S}$ or $\tau_{th} = \dfrac{R_{th} \times C_p{}^2}{\theta\tau \times S}$,

where: F_p (N) is the motor peak force, C_p (N.m) is the motor peak torque and

$\theta\tau$ (°C/s) is the temperature rise at motor peak force

- *DriverMaximumRMSCurrent*: $I_{RMS} = \min\left(\sqrt{\dfrac{\Delta T \times S}{R_{th} \times (1 + 0.004 \times \Delta t)}} \times \dfrac{\sqrt{2}}{Kt}, I_{\max}\right)$
- *DriverMaximumPeakCurrent*: $I_{peak} = \min(I_{RMS} \times r \times 1.1, I_{\max})$

Notice: the coefficient 1.1 is the margin for the servo loop transient.

- DriverRMSIntegrationTime: $\min(\tau_{th}, 15s)$
- *ScalingAcceleration:* $A_{scaling} = \dfrac{I_{\max}}{\sqrt{2}} \times \dfrac{Kt}{Load}$ (m/s²) or $A_{scaling} = \dfrac{I_{\max}}{\sqrt{2}} \times \dfrac{Kt}{2 \times \pi \times J \times G}$ (units/s²)

AccelerationLimit: $A_{limit} = \min\left(\dfrac{I_{RMS}}{\sqrt{2}} \times r \times \dfrac{Kt}{Load}, A_{scaling}\right)$ (m/s²) or

$A_{limit} = \min\left(\dfrac{I_{RMS}}{\sqrt{2}} \times r \times \dfrac{Kt}{2 \times \pi \times J \times G}, A_{scaling}\right)$ (units/s²)

5.4.5 Thermistor Threshold

Stages.ini file entry: *DriverThermistanceThreshold*

The thermistor threshold, *DriverThermistanceThreshold* (Ω), sets the threshold for the driver fault in the event of over heating. This value must be greater than 100 Ω and less than or equal to 9 kΩ.

For a 1 kΩ PTC temperature sensor, the thermistor threshold value is 1000, corresponding to the resistor value of the transition edge. The temperature that corresponds to that threshold is determined by the type of sensor (see color code of the wire).

5.5 XPS-DRV03/XPS-DRV03H with Tachometer Feedback

Stages.ini file entry:

DriverName = XPS-DRV03 ; for DRV03 driver cards

or

DriverName = XPS-DRV03H ; for DRV03H driver cards

This type of motor driver model is used for DC motors with tachometer. The driver command interface must be set to *velocity control* (cf. § 4.1) and the position servo loop type to *PID with velocity output (*cf. § 3.1). The XPS-DRV03 driver board supplies a maximum output of 5 A and 48 V. The XPS-DRV03H driver board supplies a maximum output of 1.58 A and 48 V.

5.5.1 Motor Winding Resistance

Stages.ini file entry: *DriverMotorResistance*

The motor winding resistance, *DriverMotorResistance* (Ω), is the motor resistance. This value must be greater than zero and less than or equal to 655.35 Ω.

5.5.2 Motor Winding Induction

Stages.ini file entry: *DriverMotorInductance*

The motor winding induction, *DriverMotorInductance* (H), is the motor inductance. This value must be greater than zero and less than or equal to 65.535 mH.

5.5.3 Motor Voltage Constant

Stages.ini file entry: *DriverMotorVoltageConstant*

The motor voltage constant parameter, *DriverMotorVoltageConstant* (Volt/rpm), sets the back EMF constant of the motor. This value must be greater than zero and less than or equal to 65.535e^{-3} V/rpm.

5.5.4 Tachometer Generator Voltage

Stages.ini file entry: *DriverTachoGeneratorVoltage*

The tachometer generator voltage parameter, *DriverTachoGeneratorVoltage* (Volt/rpm), set the voltage constant of the tachometer generator. This value must be greater than zero and less than or equal to 65.535e^{-3} V/rpm.

5.5.5 Stage inertia

Stages.ini file entry: *DriverStageInertia*

The stage inertia, *DriverStageInertia* (kg.m²), is the total inertia (*J*) on the motor axis. It must be greater or equal to 10^{-9} kg.m² and less than or equal to 1 kg.m².

Notice: $J = J_{motor} + J_{load} + J_{mec}$

with J_{motor}: motor rotor inertia (kg.m²)

J_{load}: load inertia (kg.m²)

J_{mec}: bearing, lead screw, … inertia (kg.m²)

5.5.6 Gear Ratio

Stage.ini file entry: *DriverGearRatio*

The gear ratio, *DriverGearRatio* (revolution/unit), sets the ratio between the motor rotation and the stage displacement. It must be greater than 0.

5.5.7 Current Servo Loop Cut Off Frequency

Stages.ini file entry: *DriverCurrentCutOffFrequency*

The current servo loop cut off frequency, *DriverCurrentCutOffFrequency* (Hz), sets the bandwidth of the internal driver current servo loop. The driver internal corrector parameters are calculated automatically. This value has to be set in relation to the bandwidth of the velocity servo loop (cf. next §). This value must be greater than zero and less than or equal to 3 kHz.

For a stage with a velocity servo loop cut off frequency between 100 and 200 Hz, the current servo loop cut off frequency should be set between 500 and 1000 Hz.

5.5.8 Velocity Servo Loop Cut Off Frequency

Stages.ini file entry: *DriverVelocityCutOffFrequency*

The velocity servo loop cut off frequency, *DriverVelocityCutOffFrequency* (Hz), sets the bandwidth of the internal driver velocity servo loop. The driver internal corrector parameters are calculated automatically. This value has to be set in relation to the bandwidth of the velocity servo loop (cf. § 3.1.1). This value must be greater than zero and less than or equal to 300 Hz.

For a stage with a velocity servo loop cut off frequency between 20 and 50 Hz, the velocity servo loop cut off frequency should be set between 100 and 200 Hz.

5.5.9 Current Monitoring Parameters

Needed entries in the configuration file:

- RMS current limit: *DriverMaximumRMSCurrent*
- RMS integration time: *DriverRMSIntegrationTime*

The RMS current limit, *DriverMaximumRMSCurrent* (A), is the maximum allowed RMS motor current. The RMS integration time *DriverRMSIntegrationTime* (s) defines the integration time span. If the RMS motor current goes beyond this value, a driver fault is generated. The RMS Current limit must be greater than zero and less than or equal to 5 A. The RMS integration time must be greater than zero and less than or equal to 60 s.

There are many different methods to define these values. Here is a description of a method used by Newport for standard products.

Application input:

- Due to the high accuracy requirements of Newport testing, a maximum temperature increase of the motor coils by 20 °C is set to avoid thermal effects impacting the motion precision: $\Delta T = 20°C$
- A ratio of 2 between the current limit (defined in the motor interface section) and the RMS current limit is used. This has been found to be a good approach in numerous precision motion applications offering a good balance between throughput and precision: $r = 2$

Motor data

- motor torque constant: Kt (N.m/Arms)
- motor constant or steepness: S (N².m²/W)
- thermal resistance: R_{th} (°C/W)
- thermal time constant: τ_{th} (s)

Driver data

- maximum driver current: $I_{max} = 5A$

Calculations:

- $Kt = \sqrt{Rf \times S}$,

where Rf (Ω) is the motor resistance per phase

- $\tau_{th} = \frac{R_{th} \times C_p{}^2}{\theta\tau \times S}$,

where: C_p (N.m) is the motor peak torque and

$\theta\tau$ (°C/s) is the temperature rise at motor peak torque

- *DriverMaximumRMSCurrent*: $I_{RMS} = \min\left(\sqrt{\frac{\Delta T \times S}{R_{th} \times (1 + 0.004 \times \Delta t)}} \times \frac{1}{Kt}, I_{max}\right)$
- $CurrentLimit = \min(I_{RMS} \times r, I_{max})$
- DriverRMSIntegrationTime: $\min(\tau_{th}, 3s)$

5.5.10 Maximum Allowed Motor Voltage

Stages.ini file entry: *DriverMaximumMotorVoltage*

The maximum allowed motor voltage, *DriverMaximumMotorVoltage* (V), sets the maximum allowed output voltage of the driver. It must be greater than zero and less than or equal to 48 V.

This parameter can be determined as follows:

Motor data

- motor winding resistance per phase: R_{mot} (Ω)
- motor torque constant: Kt (N.m/A)
- motor voltage constant: Kv (V/rpm)
- maximum allowed motor current: *MotorCurrentLimit* (A)
- maximum allowed motor voltage: *MotorVoltageLimit* (V)

Driver data

motor current at maximum command: ScalingCurrent (A) (5A for XPS-DRV03)

motor voltage at maximum command; ScalingVoltage (V) (48 V for XPS-DRV03)

Stage data:

- ratio between motor rotation and stage displacement: G (revolution/units)
- total inertia on the motor axis: J (kg.m²)

Notice: $J = J_{motor} + J_{load} + J_{mec}$

with J_{motor}: motor rotor inertia (kg.m²)

J_{load}: load inertia (kg.m²)

J_{mec}: bearing, lead screw, … inertia (kg.m²)

User performance:

- maximum stage velocity (cf. § 7.2.4): $MaximumVelocity$ (units/s)
- maximum stage acceleration (cf. § 7.2.5): $MaximumAcceleration$ (units/s²)

Maximum allowed motor voltage:

- $MaximumCurrent = \min\left(\frac{MaximumAcceleration \times J \times 2 \times \pi \times G}{Kt}, MotorCurrentLimit, ScalingCurrent\right)$
- $MaximumVoltage = R_{mot} \times MaximumCurrent + MaximumVelocity \times 60 \times G \times Kv$
- $VoltageLimit = \min(MaximumVoltage \times 1.5, MotorVoltageLimit, ScalingVoltage)$

NOTE

It is recommended that the VoltageLimit be 1.5 times the motion profiler maximum voltage to meet the motion requirements of the default stage dynamics.

5.6 XPS-DRV03/XPS-DRV03H for Acceleration Control

Stages.ini file entry:

DriverName = XPS-DRV03 ; for DRV03 driver cards

or

DriverName = XPS-DRV03H ; for DRV03H driver cards

This type of motor driver model is used for DC motors controlled by acceleration. The driver command interface must be set to *acceleration control* (cf. § 4.3) and the position servo loop type to *PID with acceleration output* (cf. § 3.3). The XPS-DRV03 driver board supplies a maximum output of 5 A and 48 V. The XPS-DRV03H driver board supplies a maximum output of 1.58 A and 48 V.

5.6.1 Motor Winding Resistance

Stages.ini file entry: *DriverMotorResistance*

The motor winding resistance, *DriverMotorResistance* (Ω), is the motor resistance. This value must be greater than zero and less than or equal to 65.535 Ω.

5.6.2 Motor Winding Inductance

Stages.ini file entry: *DriverMotorInductance*

The motor winding inductance, *DriverMotorInductance* (H), is the motor inductance. This value must be greater than zero and less than or equal to 65.535 mH.

5.6.3 Current Servo Loop Cut Off Frequency

Stages.ini file entry: *DriverCurrentCutOffFrequency*

The current servo loop cut off frequency, *DriverCurrentCutOffFrequency* (Hz), sets the bandwidth of the internal driver current servo loop. The internal driver corrector parameters are calculated automatically. This value has to be set in relation to the bandwidth of the position servo loop (cf. § 3.4.1). This value must be greater than zero and less than or equal to 3 kHz.

For a stage with a position servo loop cut off frequency between 20 and 50 Hz, the *current servo loop cut off frequency* should be set between 200 and 500 Hz.

5.6.4 Current Monitoring Parameters

Needed entries in the configuration file:

- peak current limit: *DriverMaximumPeakCurrent*
- RMS current limit: *DriverMaximumRMSCurrent*
- RMS integration time: *DriverRMSIntegrationTime*

The peak current limit, *DriverMaximumPeakCurrent* (A), is the maximum allowed motor current. If the motor current exceeds this value, a driver fault is generated. This value must be greater than zero and less than or equal to 5 A.

The RMS current limit, *DriverMaximumRMSCurrent* (A), is the maximum allowed RMS motor current. The RMS integration time *DriverRMSIntegrationTime* (s) defines the integration time span. If the RMS motor current exceeds this value, a driver fault is generated. The *RMS Current limit* must be greater than zero and less than or equal to 5 A. The *RMS integration time* must be greater than zero and less than or equal to 60 s.

There are many different methods to define these values. Here is a description of a method used by Newport for standard products.

Application input:

- Because high accuracy needs, a maximum temperature raise of the motor coils by 20 °C is set to avoid thermal effects impacting the motion precision: $\Delta T = 20^oC$
- A ratio of 2 between the peak current limit and the RMS current limit is used. This has been found to be a good approach in numerous precision motion applications offering a good balance between throughput and precision: $r = 2$

Motor data

- motor torque constant: Kt (N.m/Arms)
- motor constant or steepness: S (N²m²/W)
- thermal resistance: R_{th} (°C/W)
- thermal time constant: τ_{th} (s)

Driver data

- maximum driver current: $I_{max} = 5A$

Stage data

- ratio between motor rotation and stage displacement: G (revolution/units)
- total inertia on the motor axis: J (kg.m²)

Notice: $J = J_{motor} + J_{load} + J_{mec}$

with J_{motor}: motor rotor inertia (kg.m²)

J_{load}: load inertia (kg.m²)

J_{mec}: bearing, lead screw, … inertia (kg.m²)

Calculations:

- $Kt = \sqrt{Rf \times S}$,

where Rf (Ω) is the motor resistance per phase

- $\tau_{th} = \dfrac{R_{th} \times C_p^{\,2}}{\theta\tau \times S}$,

where: C_p (N.m) is the motor peak torque and

$\theta\tau$ (°C/s) is the temperature rise at motor peak torque

- *DriverMaximumRMSCurrent*: $I_{RMS} = \min\left(\sqrt{\dfrac{\Delta T \times S}{R_{th} \times (1 + 0.004 \times \Delta t)}} \times \dfrac{1}{Kt}, \mathrm{I}_{\max} \right)$
- *DriverMaximumPeakCurrent*: $I_{peak} = \min(I_{RMS} \times r \times 1.1, \mathrm{I}_{\max})$

Notice: the coefficient 1.1 is the margin for the servo loop transient.

- DriverRMSIntegrationTime: $\min(\tau_{th}, 3s)$
- ScalingAcceleration: $A_{scaling} = I_{\max} \times \dfrac{Kt}{J \times G \times 2\pi}$ (units/s²)
- *AccelerationLimit*: $A_{limit} = \min\left(I_{RMS} \times r \times \dfrac{Kt}{J \times G \times 2\pi}, \mathrm{A}_{scaling} \right)$ (units/s²)

5.6.5 Maximum Allowed Motor Voltage

Stages.ini file entry: *DriverMaximumMotorVoltage*

The maximum allowed motor voltage, *DriverMaximumMotorVoltage* (V), sets the maximum allowed output voltage of the driver.

5.7 XPS-DRV03 for Voltage Control

Stages.ini file entry:

DriverName = XPS-DRV03 ; for DRV03 driver cards

or

DriverName = XPS-DRV03H ; for DRV03H driver cards

This setting of the motor driver model is used for DC motors controlled directly by the motor voltage. The driver command interface must be set to *voltage control* (cf. § 4.2) and the position servo loop type to *PID with voltage output (*cf. § 3.2). The XPS-DRV03 driver board supplies a maximum output of 5 A and 48 V. The XPS-DRV03H driver board supplies a maximum output of 1.58 A and 48 V.

5.7.1 Current Monitoring Parameters

Needed entries in the configuration file:

- RMS current limit: *DriverMaximumRMSCurrent*
- RMS integration time: *DriverRMSIntegrationTime*

The RMS current limit, *DriverMaximumRMSCurrent* (A), is the maximum allowed RMS motor current. The RMS integration time *DriverRMSIntegrationTime* (s) defines the integration time span. If the RMS motor current goes beyond this value, a driver fault is generated. The *RMS Current limit* must be greater than zero and less than or equal to 5 A. The *RMS integration time* must be greater than zero and less than or equal to 60 s.

There are many different methods to define these values. Here is a description of a method used by Newport for standard products.

Application input:

- Because high accuracy needs, a maximum temperature raise of the motor coils by 20 °C is set to avoid thermal effects impacting the motion precision: $\Delta T = 20°C$
- A ratio of 2 between the current limit (defined in the motor interface section) and the RMS current limit is used. This has been found to be a good approach in numerous precision motion applications offering a good balance between throughput and precision: $r = 2$

Motor data

- motor torque constant: Kt (N.m/Arms)
- motor constant or steepness: S ($N^2.m^2/W$)
- thermal resistance: R_{th} (°C/W)
- thermal time constant: τ_{th} (s)

Driver data

- maximum driver current: $I_{max} = 5A$

Calculations:

- $Kt = \sqrt{Rf \times S}$,

where Rf (Ω) is the motor resistance per phase

- $\tau_{th} = \frac{R_{th} \times C_p^{\ 2}}{\theta\tau \times S}$,

Where: C_p (N.m) is the motor peak torque and

$\theta\tau$ (°C/s) is the temperature rise at motor peak torque

- *DriverMaximumRMSCurrent*: $I_{RMS} = \min\left(\sqrt{\frac{\Delta T \times S}{R_{th} \times (1 + 0.004 \times \Delta t)}} \times \frac{1}{Kt}, I_{max}\right)$
- $CurrentLimit = \min(I_{RMS} \times r, I_{max})$
- DriverRMSIntegrationTime: $\min(\tau_{th}, 3s)$

5.8 XPS-DRV00 for Non-Configurable External Driver

Stages.ini file entry: *DriverName = XPS-DRV00*

The XPS-DRV00 is a pass-through board which is necessary when connecting the XPS controller to an external motor driver. This setting of the motor driver model is compatible with all driver command interfaces.

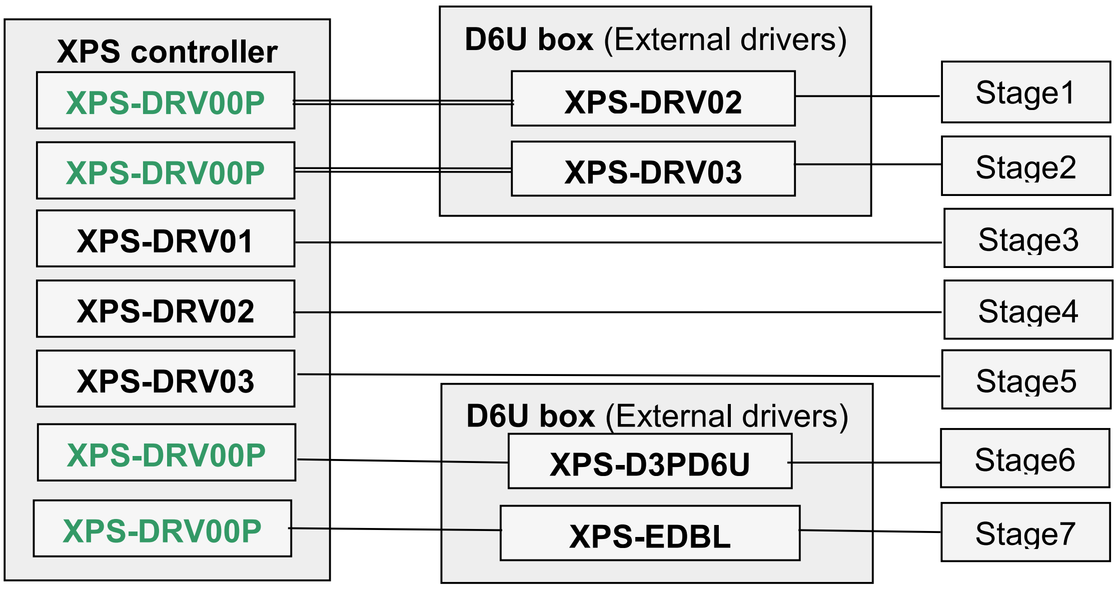
5.9 XPS-DRV00P for Configurable External Driver

Stages.ini file entry: *DriverName = XPS-DRV00P*

The XPS-DRV00P (DRV00 version 2) is a pass through board developed for the CIE05/CIE08 board. It is an interconnect board for external amplifiers connected to the XPS controller. The XPS-DRV00P is an XPX-DRV00 card, but added with Pulses/Direction outputs for stepper motor control (Pulse/Dir or Pulse+/Pulse- mode), with an I2C communication link allowing the user to configure the configurable external drivers (for example external XPS-DRV02, XPS-DRV03, XPS-D3PD6U or XPS-EDBL Newport drivers).

How to use a configurable external driver?

A configurable external driver must be connected to the XPS controller with an XPS-DRV00P board.



NOTE

In the stages.ini file, the "DriverName" of your stage configuration must be declared as XPS-DRV02, XPS-DRV03, XPS-D3PD6U or XPS-EDBL, but not XPS-DRV00P.

```
[Stage1]
DriverName = XPS-DRV02
[Stage2]
DriverName = XPS-DRV03
[Stage6]
DriverName = XPS-D3PD6U
[Stage7]
DriverName = XPS-EDBL
```

5.10 NON_CONFIGURABLE_DRV for Directly Connected Non-Configurable External Driver

Stages.ini file entry: *DriverName = NON_CONFIGURABLE_DRV*

This type of DriverName is used when connecting the XPS controller to a non-configurable external motor driver by a cable without use of a DRV00/DRV00P pass-through card. This setting of the motor driver model is compatible with all driver command interfaces.

5.11 XPS-DRVPx (x = 1, 2, …) for Piezo Driver

Stages.ini file entry: *DriverName = XPS-DRVPx (x = 1, 2, ...)*

This type of DriverName is used with a piezo driver card. This setting of the motor driver model is compatible only with

AnalogPositionPiezo driver command interfaces,

NoEncoderPosition or PIPosition corrector type.

Driver parameters

DriverNotchFrequency

DriverNotchBandwidth

DriverNotchGain

DriverLowpassFrequency

DriverKI

DriverFatalFollowingError

DriverStagePositionOffset

DriverTravelCorrection

6.0 Position Encoder Interface

The XPS controller supports 2 types of position encoder interfaces:

- AquadB differential
- AquadB differential with sine/cosine 1 Vpp
- AquadB differential THETA
- AquadB differential THETA with sine/cosine 1 Vpp THETA

6.1 RS422 Differential (AquadB)

Stages.ini file entry: *EncoderType = AquadB*

This encoder type is used when the position sensor delivers two square waves RS422 A differential signals.

6.1.1 Stage Displacement per Encoder Count

Stages.ini file entry: *EncoderResolution*

The stage displacement per encoder count, *EncoderResolution* (units), sets the resolution of the position encoder. It must be greater than zero.

NOTE

The encoder resolution is equal to 4-times the signal period (quadrature effect).

The *Stage displacement per encoder count* essentially defines the measurement units of the stage. Many parameters are derived from this value, in particular all parameters with units of length, such as velocities or accelerations. Therefore, it is critical this parameter be set correctly for proper operation.

To calculate the *Stage displacement per encoder count* value all of the following must be taken into account: the number of steps per revolution, screw pitch and any gear reduction in the stage.

6.1.2 Linear Correction

Stages.ini file entry: *LinearEncoderCorrection*

LinearEncoderCorrection = ((Real increment/Rounded increment) - 1) * 1e^{6}

The linear correction, *LinearEncoderCorrection* (parts per million), sets the correction applied to the *EncoderResolution* to compensate for linear error effects (see chapter “Compensation/Linear Error Correction” in the XPS Motion Tutorial). This value must be between ±0.5*10^{6}. A zero value disables this feature.

$$CorrectedResolution = (1 + LinearCorrection \times 10^{-6}) \times EncoderResolution$$

The default value is zero.

6.1.3 Stage Backlash

Stages.ini file entry: *Backlash*

The stage backlash, *Backlash* (units), sets the backlash compensation value applied to the target position for a move (see chapter “Compensation/Backlash Compensation” in the XPS Motion Tutorial). This value must be greater than or equal to zero (backlash disabled).

The default value is zero.

6.1.4 Gathering Cut Off Frequencies

Needed entries in the configuration file:

- gathering velocity cut off frequency: *CurrentVelocityCutOffFrequency*
- gathering acceleration cut off frequency: *CurrentAccelerationCutOffFrequency*

The gathering cut off frequencies, *CurrentVelocityCutOffFrequency* (Hz) and *CurrentAcceleration-CutOffFrequency* (Hz), set the cut off frequencies for the low-pass filters that get applied to the CurrentVelocity and CurrentAcceleration saved by the gathering feature. This filter reduces the derivative noise. These values must be greater than zero (filter disabled) and less than half of the servo loop frequency (8 kHz).

The default value is 100 Hz which is about five times greater than the bandwidth of the position servo loop of a typical screw driven stage.

6.1.5 Positioner mapping parameters

Needed entries in the configuration file:

- file name: *PositionerMappingFileName*
- line number: *PositionerMappingLineNumber*
- maximum position error: *PositionerMappingMaxPositionError*

The *PositionerMappingFileName* defines the name of the mapping file used for the positioner mapping compensation. No entry for the file name disables the feature.

The *PositionerMappingLineNumber* defines the number of data lines in the positioner mapping file. This value must be greater than or equal to 3 and less than 200. This parameter is primarily used as a check to confirm the correctness of the mapping file.

The *PositionerMappingMaxPositionError* (units) defines the maximum absolute value of the error corrections in the mapping file. This parameter is primarily used as a check to confirm the correctness of the mapping file.

Please refer to the XPS Motion Tutorial chapter “Compensation/Positioner Mapping” for further information about the positioner mapping functionality.

6.2 RS422 Differential with 3 Encoders (AquadBTheta)

Stages.ini file entry: *EncoderType = AquadBTheta*

This encoder type is composed of three encoders and is used when the position sensor delivers two square waves RS422 A differential signals.

NOTE

Same parameters as required with an AquadB encoder with additional parameters required.

6.2.1 Theta Radius

Required entries in the configuration file:

Theta encoder radius: EncoderRadius

The encoder radius, *EncoderRadius (XY *10-6),* is the radius of the theta (three encoders)

6.2.2 X and Y Correction Limits

Required entries in the configuration file:

Limit of the correction X: MaximumEncoderCorrectionX

Limit of the correction Y: MaximumEncoderCorrectionY

The *MaximumEncoderCorrectionX* sets the maximum allowed correction for the X positioner.

The *MaximumEncoderCorrectionY* sets the maximum allowed correction for the Y positioner.

6.3 Sine/Cosine 1Vpp (AnalogInterpolated)

Stages.ini file entry: *AnalogInterpolated*

This encoder type is used when the position sensor delivers two 1 Vpp sine/cosine signals that are interpolated by the XPS controller.

6.3.1 Resolution

Required entries in the configuration file:

- stage displacement per encoder period: *EncoderScalePitch*
- Encoder signal subdivisor: *EncoderInterpolationFactor*

The stage displacement per encoder period, *EncoderScalePitch* (units), sets the displacement of the stage per encoder period. This parameter essentially defines the measurement units of the stage. Many parameters are derived from this entry, in particular all parameters with units of length, such as velocities or accelerations. Therefore it is critical this parameter value is set correctly for proper operation of the stage.

The Encoder signal subdivisor, *EncoderInterpolationFactor*, sets the interpolation factor for the encoder signal interpolation. This value defines the resolution of the CurrentPosition and SetpointPosition of the stage as follows:

Resolution = EncoderScalePitch/EncoderInterpolationFactor.

For a better understanding of the meaning of the CurrentPosition and SetpointPosition see the XPS Motion Tutorial, chapter on natural units.

The *EncoderInterpolationFactor* must be an integer value greater than or equal to 1 and less than or equal to 32768.

The *EncoderInterpolationFactor* should be set relative to the position noise of the stage. It is not recommended to set a position resolution that is far less than the amplitude of the position noise. It is also important to note that the value of the *EncoderInterpolationFactor* has no impact on the resolution of the position used for the position servo loop. The position servo loop always uses calculations at the maximum position resolution possible. For example, (EncoderScalePitch/32768).

6.3.2 Offsets Correction

Needed entries in the configuration file:

- sine channel offset correction: *EncoderSinusOffset*
- sine channel offset correction: *EncoderCosinusOffset*

The sine/cosine channel offset corrections, *EncoderSinusOffset* (V) and *EncoderCosinusOffset* (V), set the offset values of the two encoder signals for correction. They must be between ±0.1 V. A zero value disables this feature.

The default value is zero.

6.3.3 Amplitude Correction

Stages.ini file entry: *EncoderDifferentialGain*

The amplitude correction, *EncoderDifferentialGain*, sets the amplitude correction of the cosine signal amplitude. It must be between ±0.1. A zero value disables this feature.

$$CorrectedCosine = (1 + EncoderDifferentialGain) \times Cosine$$

The default value is zero.

6.3.4 Phase Correction

Stages.ini file entry: *EncoderPhaseCompensation*

The signal phase correction, *EncoderPhaseCompensation* (°), sets the phase correction of the cosine signal phase. This correction is applied after the differential amplitude correction. This value must be between ±10 °. A zero value disables this feature.

$$PhaseCorrectedCosine = \frac{Sine \times \sin\left(PhaseCompensation \times \frac{\pi}{180}\right) + CorrectedCosine}{\cos\left(PhaseCompensation \times \frac{\pi}{180}\right)}$$

The default value is zero.

6.3.5 Mechanical Zero Input Plug

Stages.ini file entry: *EncoderZMPlug*

The mechanical zero input plug, *EncoderZMPlug*, defines where the mechanical zero signal is input. There are two possible settings:

Driver	for mechanical zero through the driver board plug
Encoder	for mechanical zero through the encoder driver board plug.

6.3.6 Linear Correction

Stages.ini file entry: *LinearEncoderCorrection*

LinearEncoderCorrection = ((Real increment/Rounded increment) - 1) * e^6

The linear correction, *LinearEncoderCorrection* (parts per million), sets the correction applied to the *EncoderResolution* to compensate for linear error effects (see chapter “Compensation/Linear Error Correction” in the XPS Motion Tutorial). This value must be between ±0.5*10^6. A zero value disables this feature.

$$Corrected\,\mathrm{Re}solution = (1 + LinearCorrection \times 10^{-6}) \times Encoder\,\mathrm{Re}solution$$

The default value is zero.

6.3.7 Stage Backlash

Stages.ini file entry: *Backlash*

The stage backlash, *Backlash* (units), sets the backlash compensation value applied to the target position for a move (see chapter “Compensation/Backlash Compensation” in the XPS Motion Tutorial). This value must be greater than or equal to zero (backlash disabled).

The default value is zero.

6.3.8 Gathering Cut Off Frequencies

Needed entries in the configuration file:

- gathering velocity cut off frequency: *CurrentVelocityCutOffFrequency*
- gathering acceleration cut off frequency: *CurrentAccelerationCutOffFrequency*

The gathering cut off frequencies, *CurrentVelocityCutOffFrequency* (Hz) and *CurrentAcceleration-CutOffFrequency* (Hz), set the cut off frequencies for low-pass filters that are applied to the CurrentVelocity and CurrentAcceleration saved by the gathering feature. These filters reduce derivative noise. They must be greater than zero (filter disabled) and less than half of the servo loop frequency (8 kHz).

The default value is 100 Hz which is about five times greater than the bandwidth of the position servo loop of a typical screw driven stage.

6.3.9 Positioner Mapping Parameters

Configuration file entries:

- file name: *PositionerMappingFileName*
- line number: *PositionerMappingLineNumber*
- maximum position error: *PositionerMappingMaxPositionError*

The *PositionerMappingFileName* defines the name of the mapping file used for the positioner mapping compensation. No entry for the file name disables the feature.

The *PositionerMappingLineNumber* defines the number of data lines in the positioner mapping file. This value must be greater than or equal to 3 and less than 200. This parameter is primarily used as a check to confirm the correctness of the mapping file.

The *PositionerMappingMaxPositionError* (units) defines the maximum absolute value of the error corrections in the mapping file. This parameter is primarily used as a check to confirm the correctness of the mapping file.

Please refer to the XPS Motion Tutorial, chapter "Compensation/Positioner Mapping" for further information about the positioner mapping functionality.

6.4 Sine/Cosine 1 Vpp (AnalogInterpolatedTheta)

Stages.ini file entry: *EncoderType = AnalogInterpolatedTheta*

This encoder type is composed of three encoders and is used when the position sensor delivers two 1Vpp sine/cosine signals that get interpolated by the XPS controller.

NOTE

These parameters are the same as for *AnalogInterpolated* Encoder, with additions.

6.4.1 Theta Radius

Needed entries in the configuration file:

Theta encoder radius: EncoderRadius

The encoder radius, *EncoderRadius (XY *10^{-6}),* is the radius of the theta (three encoders)

6.4.2 X and Y Correction Limits

Needed entries in the configuration file:

Limit of the correction X: MaximumEncoderCorrectionX

Limit of the correction Y: MaximumEncoderCorrectionY

The *MaximumEncoderCorrectionX* sets the maximum allowed correction for the X positioner.

The *MaximumEncoderCorrectionY* sets the maximum allowed correction for the Y positioner.

7.0 Limit Sensors Input Plug

The XPS controller supports 3 settings for the limit sensors input plug:

- Driver board
- Encoder board
- None for spindle group
- None for piezo driver

The choice of the setting for the limit sensor input plug depends on where these signals are electrically connected to the XPS controller: Through the plug on the driver board or through the plug on the encoder board. The setting *none for spindle group* is only used for spindle groups.

The limit sensor input plug setting is made with other stage settings, such as: maximum stage travel, maximum commandable speeds and accelerations, etc.

7.1 Driver Board

Stages.ini file entry: *ServitudesType = StandardEORDriverPlug*

This setting is used when the travel limit signals are wired to the XPS controller through the driver board plug.

7.1.1 Minimum Position

Stages.ini file entry: *MinimumTargetPosition*

The minimum position, *MinimumTargetPosition* (units), sets the minimum allowed position for any move command. This value must be less than the maximum position.

7.1.2 Maximum Position

Stages.ini file entry: *MaximumTargetPosition*

The maximum position, *MaximumTargetPosition* (units), sets the maximum allowed position for any move command. This value must be greater than the Home Preset.

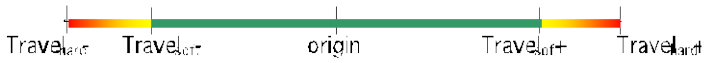


Figure 32: Minimum and maximum travel positions

7.1.3 Home Position

Stages.ini file entry: *HomePreset*

The home position, *HomePreset* (units), sets the position value of the home reference. This value must be between the minimum position (cf. § 7.2.1) and the maximum position (cf. § 7.2.2).

7.1.4 Maximum Velocity

Stages.ini file entry: *MaximumVelocity*

The maximum velocity, *MaximumVelocity* (units/s), sets the maximum velocity the stages can be commanded to move. This value must be greater than zero and less than or equal to the stage velocity at maximum command (cf. § 4.1.1).

When used with stepper motors and sine/cosine position control the value for the *maximum velocity* must be less than or equal to the *DisplacementPerFullStep* (cf. §4.4.2) multiplied by the servo loop frequency (8 kHz). For smooth operation, however, we recommend to not exceed one quarter of this maximum possible value.

7.1.5 Maximum Acceleration

Stages.ini file entry: *MaximumAcceleration*

The maximum acceleration, *MaximumAcceleration* (units/s²), sets the maximum acceleration the stage can be commanded to move. This value must be greater than zero and less than or equal to the stage acceleration at maximum command (cf. § 4.3.1).

The recommended value for smooth displacement is four times the maximum velocity.

7.1.6 SGamma Profile Generator Jerk Times

Required entries in the configuration file:

- SGamma profile generator minimum jerk time: *MinimumJerkTime*
- SGamma profile generator maximum jerk time: *MaximumJerkTime*

The SGamma profile generator jerk times, *MinimumJerkTime* (s) and *MaximumJerkTime* (s), set the jerk times of the SGamma profile generator. The minimum jerk time must be less than the maximum jerk time and greater than or equal to the profile generator cycle period (0.4 ms).

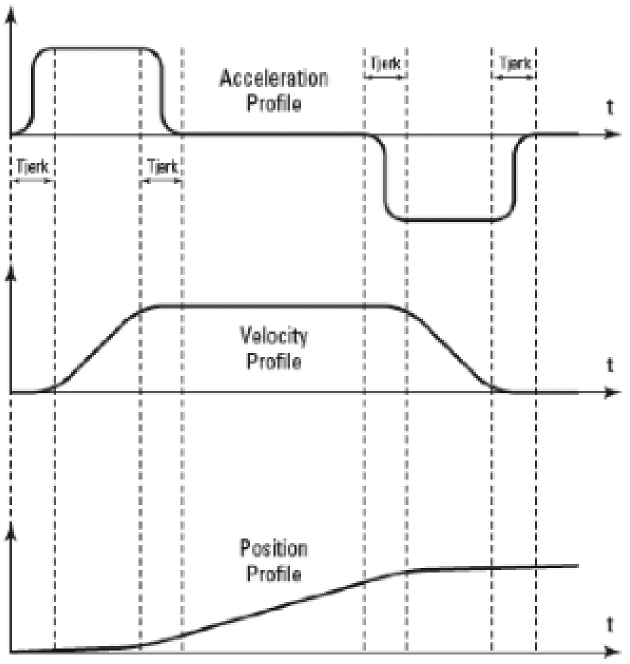


Figure 33: SGamma motion profile.

The recommended values for smooth displacement are:

- $\text{MaximumJerkTime} = 0.05s$
- $\text{MinimumJerkTime} = 0.005s$

For more information about the SGamma motion profiler, see also the XPS Motion Tutorial, chapter "Motion Profiles".

7.1.7 Emergency Deceleration Multiplier

Stages.ini file entry: *EmergencyDecelerationMultiplier*

The emergency deceleration multiplier, *EmergencyDecelerationMultiplier*, defines the ratio between the emergency deceleration during an emergency brake and the normal deceleration during normal brake. This value must be greater than or equal to one.

The default setting is 4.

7.1.8 Tracking Mode Filter Cut Off Frequency

Stages.ini file entry: *TrackingCutOffFrequency*

The tracking mode filter cut off frequency, *TrackingCutOffFrequency* (Hz), sets the cut off frequency of the filter applied to the tracking input. This frequency must be at least ten times less than the servo loop bandwidth to have a proper tracking. This value must be also greater than or equal to zero (disable) and less than or equal to half of the profile generator cycle frequency (2500 Hz).

The default setting is 5 Hz.

7.2 Encoder Board

This setting is used when the travel limit and home signals are wired to the XPS controller through the encoder board plug.

Stages.ini file entries (for CIE08 cards and further):

1. All travel limit and home signals are wired to the XPS controller through the encoder board plug:

   ServitudesType = StandardLimitAndHomeEncoderPlug (obsolete name: StandardEOREncoderPlug)

2. Only travel limit signals are wired to the XPS controller through the encoder board plug:

   ServitudesType = StandardLimitAndLimitEncoderPlug

7.2.1 Minimum Position

Stages.ini file entry: *MinimumTargetPosition*

The minimum position, *MinimumTargetPosition* (units), sets the minimum allowed position for any move command. This value must be less than the maximum position.

7.2.2 Maximum Position

Stages.ini file entry: *MaximumTargetPosition*

The maximum position, *MaximumTargetPosition* (units), sets the maximum allowed position for any move command. This value must be greater than the Home Preset.

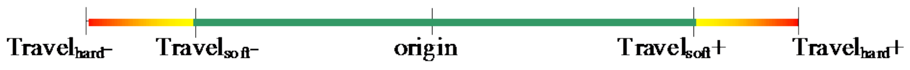


Figure 34: Minimum and maximum travel positions.

7.2.3 Home Position

Stages.ini file entry: *HomePreset*

The home position, *HomePreset* (units), sets the position value of the home reference. This value must be between the minimum position (cf. § 7.2.1) and the maximum position (cf. § 7.2.2).

7.2.4 Maximum Velocity

Stages.ini file entry: *MaximumVelocity*

The maximum velocity, *MaximumVelocity* (units/s), sets the maximum velocity the stages can be commanded to move. This value must be greater than zero and less than or equal to the stage velocity at maximum command (cf. § 4.1.1).

When used with stepper motors and sine/cosine position control the value for the *maximum velocity* must be less than or equal to the *DisplacementPerFullStep* (cf.

§4.4.2) multiplied by the servo loop frequency (8 kHz). For smooth operation, however, we recommend not to exceed one quarter of this maximum possible value.

7.2.5 Maximum Acceleration

Stages.ini file entry: *MaximumAcceleration*

The maximum acceleration, *MaximumAcceleration* (units/s^2), sets the maximum acceleration the stage can be commanded to move. This value must be greater than zero and less than or equal to the stage acceleration at maximum command (cf. § 4.3.1).

The recommended value for smooth displacement is four times the maximum velocity.

7.2.6 SGamma Profile Generator Jerk Times

Configuration file entries:

- SGamma profile generator minimum jerk time: *MinimumJerkTime*
- SGamma profile generator maximum jerk time: *MaximumJerkTime*

The SGamma profile generator jerk times, *MinimumJerkTime* (s) and *MaximumJerkTime* (s), set the jerk times of the SGamma profile generator. The minimum jerk time must be less than the maximum jerk time and greater than or equal to the profile generator cycle period (0.4 ms).

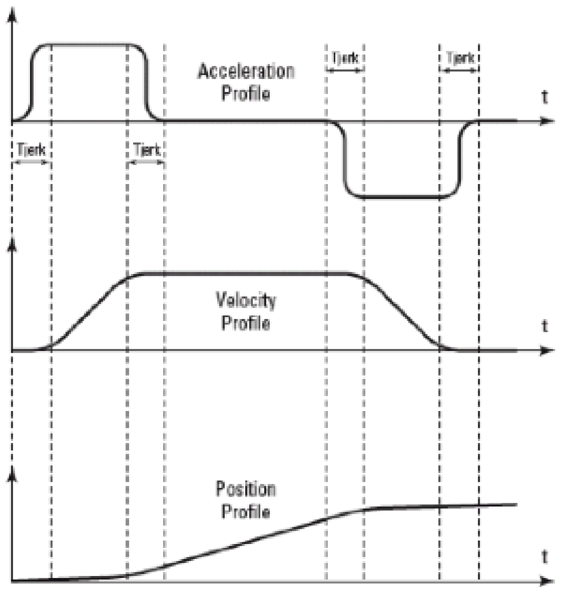


Figure 35: SGamma motion profile.

The recommended values for a smooth displacement are:

- $\text{MaximumJerkTime} = 0.05s$
- $\text{MinimumJerkTime} = 0.005s$

For more information about the SGamma motion profiler, see also the XPS Motion Tutorial, chapter “Motion Profiles”.

7.2.7 Emergency Deceleration Multiplier

Stages.ini file entry: *EmergencyDecelerationMultiplier*

The emergency deceleration multiplier, *EmergencyDecelerationMultiplier*, defines the ratio between the emergency deceleration during an emergency brake and the normal deceleration during normal brake. It must be greater than or equal to one.

The default setting is 4.

7.2.8 Tracking Mode Filter Cut Off Frequency

Stages.ini file entry: *TrackingCutOffFrequency*

The tracking mode filter cut off frequency, *TrackingCutOffFrequency* (Hz), sets the cut off frequency of the filter applied to the tracking input. This frequency must be at least ten times less than the servo loop bandwidth to avoid poor tracking. It must be also greater than or equal to zero (disable) and less than or equal to half of the profile generator cycle frequency (2500 Hz).

The default setting is 5 Hz.

7.3 None for Spindle Group

Stages.ini file entry: *ServitudesType = Spindle*

This setting can only be used with stages configured as spindle groups. It disregards the end of run detection, even if the signals are present.

7.3.1 Spindle Period

Stages.ini file entry: *SpindlePeriod*

The *SpindlePeriod* (units) sets the period of the spindle rotation. This value must be greater than zero.

7.3.2 Home Position

Stages.ini file entry: *HomePreset*

The home position, *HomePreset* (units), sets the position value of the home reference. This value must be greater than zero and less than the *SpindlePeriod* (cf. § 7.3.1).

7.3.3 Maximum Velocity

Stages.ini file entry: *MaximumVelocity*

The maximum velocity, *MaximumVelocity* (units/s), sets the maximum velocity the stages can be commanded to move. This value must be greater than zero and less than or equal to the stage velocity at maximum command (cf. § 4.1.1).

When used with stepper motors and sine/cosine position control the value for the *maximum velocity* must be less than or equal to the *DisplacementPerFullStep* (cf. §4.4.2) multiplied by the servo loop frequency (8 kHz). For smooth operation, however, we recommend not to exceed one quarter of this maximum possible value.

7.3.4 Maximum Acceleration

Stages.ini file entry: *MaximumAcceleration*

The maximum acceleration, *MaximumAcceleration* (units/s^2), sets the maximum acceleration the stage can be commanded to move. This value must be greater than zero and less than or equal to the stage acceleration at maximum command (cf. § 4.3.1).

The recommended value for smooth displacement is four times the maximum velocity.

7.3.5 SGamma Profile Generator Jerk Times

Configuration file entries:

- SGamma profile generator minimum jerk time: *MinimumJerkTime*
- SGamma profile generator maximum jerk time: *MaximumJerkTime*

The SGamma profile generator jerk times, *MinimumJerkTime* (s) and *MaximumJerkTime* (s), set the jerk times of the SGamma profile generator. The minimum jerk time must be less than the maximum jerk time and greater than or equal to the profile generator cycle period (0.4 ms).

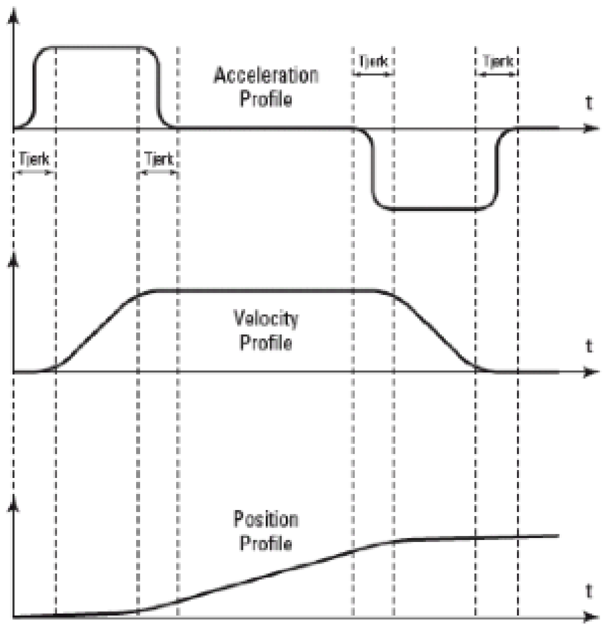


Figure 36: SGamma motion profile.

The recommended values for a smooth displacement are:

- $\text{MaximumJerkTime} = 0.05s$
- $\text{MinimumJerkTime} = 0.005s$

For more information about the SGamma motion profiler, see also the XPS Motion Tutorial, chapter "Motion Profiles".

7.3.6 Emergency Deceleration Multiplier

Stages.ini file entry: *EmergencyDecelerationMultiplier*

The emergency deceleration multiplier, *EmergencyDecelerationMultiplier*, defines the ratio between the emergency deceleration during an emergency brake and the normal deceleration during normal brake. This value must be greater than or equal to one.

The default setting is 4.

7.3.7 Tracking Mode Filter Cut Off Frequency

Stages.ini file entry: *TrackingCutOffFrequency*

The tracking mode filter cut off frequency, *TrackingCutOffFrequency* (Hz), sets the cut off frequency of the filter applied to the tracking input. This frequency must be at least ten times less than the servo loop bandwidth for proper tracking. This value must be also greater than or equal to zero (disable) and less than or equal to half of the profile generator cycle frequency (2500 Hz).

The default setting is 5 Hz.

7.4 None for Piezo Driver

Stages.ini file entry: *ServitudesType = Piezo*

This setting can only be used with piezo stages. It disregards the end of run detection, even if the signals are present.

8.0 Home Search Process

The XPS controller supports 7 different home search processes:

- mechanical zero only
- mechanical zero and index
- minus end of run only
- minus end of run and index
- current position as home
- plus end of run only
- index only

The choice depends on the availability of reference signals such as: a mechanical zero signal, a minus end of run signal or an index signal.

For more information about the possible home search processes, please refer to the XPS Motion Tutorial, chapter "Home Search".

8.1 Mechanical Zero Only

Stages.ini file entry: *HomeSearchSequenceType = MechanicalZeroHomeSearch*

This home search process is used only when the mechanical zero signal is used for the home search.

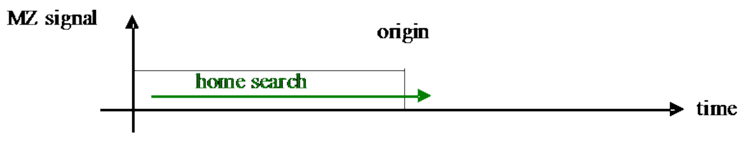


Figure 37: Mechanical zero home search process.

8.1.1 Maximum Velocity

Stages.ini file entry: *HomeSearchMaximumVelocity*

The home search maximum velocity, *HomeSearchMaximumVelocity* (units/s), sets the maximum velocity of the profile generator during the home search process. It must be greater than zero and less than or equal to the *MaximumVelocity* (cf. § 7.2.4).

The default value is one half of the *MaximumVelocity*.

8.1.2 Maximum Acceleration

Stages.ini file entry: *HomeSearchMaximumAcceleration*

The home search maximum acceleration, *HomeSearchMaximumAcceleration* (units/s^2), sets the maximum acceleration of the profile generator during the home search process. This value must be greater than zero and less than or equal to the *MaximumAcceleration* (cf. § 7.2.5).

The default value is one half of the maximum acceleration.

8.1.3 Time Out

Stages.ini file entry: *HomeSearchTimeOut*

The home search time out, *HomeSearchTimeOut* (s), sets the time out value for the home search process. When the *HomeSearchTimeOut* time is elapsed and the home search process is not yet finished, the XPS controller will generate an emergency stop and will abort the home search process. The *HomeSearchTimeOut* value must be greater than or equal to the profile generator period (400 µs).

8.2 Mechanical Zero and Index

Stages.ini file entry:

HomeSearchSequenceType = MechanicalZeroAndIndexHomeSearch

This home search process is used when both the mechanical zero signal and the index signal are used for the home search.

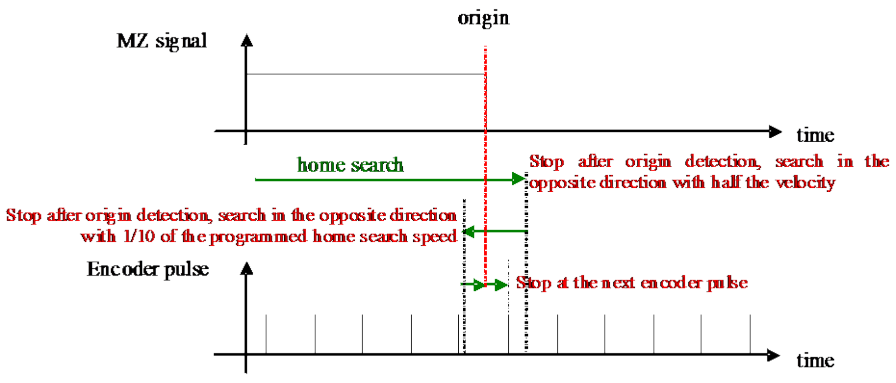

origin
MZ signal
time
home search
Stop after origin detection, search in the opposite direction with half the velocity
Stop after origin detection, search in the opposite direction with 1/10 of the programmed home search speed
Encoder pulse
Stop at the next encoder pulse
time

Figure 38: Mechanical zero and index home search process.

8.2.1 Maximum Velocity

Stages.ini file entry: *HomeSearchMaximumVelocity*

The home search maximum velocity, *HomeSearchMaximumVelocity* (units/s), sets the maximum velocity of the profile generator during the home search process. This value must be greater than zero and less than or equal to the *MaximumVelocity* (cf. § 7.2.4).

The default value is one half of the *MaximumVelocity*.

8.2.2 Maximum Acceleration

Stages.ini file entry: *HomeSearchMaximumAcceleration*

The home search maximum acceleration, *HomeSearchMaximumAcceleration* (units/s^2), sets the maximum acceleration of the profile generator during the home search process. This value must be greater than zero and less than or equal to the *MaximumAcceleration* (cf. § 7.2.5).

The default value is one half of the maximum acceleration.

8.2.3 Time Out

Stages.ini file entry: *HomeSearchTimeOut*

The home search time out, *HomeSearchTimeOut* (s), sets the time out value for the home search process. When the *HomeSearchTimeOut* time is elapsed and the home search process is not yet finished, the XPS controller will generate an emergency stop and will abort the home search process. The *HomeSearchTimeOut* value must be greater than or equal to the profile generator period (400 µs).

8.3 Minus End of Run Only Search Process

Stages.ini file entry: *HomeSearchSequenceType = MinusEndOfRunHomeSearch*

This home search process is used only when the minus end of run signal is used for the home search.

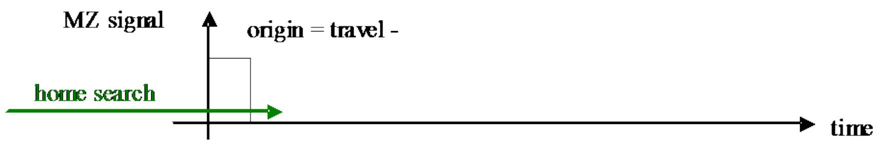


Figure 39: Minus end of run home search process.

8.3.1 Maximum Velocity

Stages.ini file entry: *HomeSearchMaximumVelocity*

The home search maximum velocity, *HomeSearchMaximumVelocity* (units/s), sets the maximum velocity of the profile generator during the home search process. This value must be greater than zero and less than or equal to the *MaximumVelocity* (cf. § 7.2.4).

The default value is one half of the *MaximumVelocity*.

8.3.2 Maximum Acceleration

Stages.ini file entry: *HomeSearchMaximumAcceleration*

The home search maximum acceleration, *HomeSearchMaximumAcceleration* (units/s^2), sets the maximum acceleration of the profile generator during the home search process. This value must be greater than zero and less than or equal to the *MaximumAcceleration* (cf. § 7.2.5).

The default value is one half of the maximum acceleration.

8.3.3 Time Out

Stages.ini file entry: *HomeSearchTimeOut*

The home search time out, *HomeSearchTimeOut* (s), sets the time out value for the home search process. When the *HomeSearchTimeOut* time is elapsed and the home search process is not yet finished, the XPS controller will generate an emergency stop and abort the home search process. The *HomeSearchTimeOut* value must be greater than or equal to the profile generator period (400 µs).

8.4 Minus End of Run and Index Search Process

Stages.ini file entry:

HomeSearchSequenceType = MinusEndOfRunAndIndexHomeSearch

This home search process is used when both the minus end of run signal and the index signal are used for the home search.

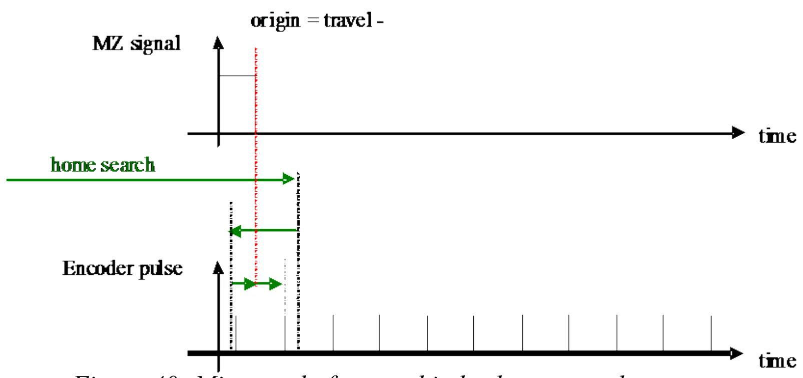


Figure 40: Minus end of run and index home search process.

8.4.1 Maximum Velocity

Stages.ini file entry: *HomeSearchMaximumVelocity*

The home search maximum velocity, *HomeSearchMaximumVelocity* (units/s), sets the maximum velocity of the profile generator during the home search process. This value must be greater than zero and less than or equal to the *MaximumVelocity* (cf. § 7.2.4).

The default value is one half of the *MaximumVelocity*.

8.4.2 Maximum Acceleration

Stages.ini file entry: *HomeSearchMaximumAcceleration*

The home search maximum acceleration, *HomeSearchMaximumAcceleration* (units/s^2), sets the maximum acceleration of the profile generator during the home search process. This value must be greater than zero and less than or equal to the *MaximumAcceleration* (cf. § 7.2.5).

The default value is one half of the maximum acceleration.

8.4.3 Time Out

Stages.ini file entry: *HomeSearchTimeOut*

The home search time out, *HomeSearchTimeOut* (s), sets the time out value for the home search process. When the *HomeSearchTimeOut* time is elapsed and the home search process is not yet finished, the XPS controller will generate an emergency stop and abort the home search process. The *HomeSearchTimeOut* value must be greater than or equal to the profile generator period (400 µs).

8.5 Current Position As Home

Stages.ini file entry:

HomeSearchSequenceType = CurrentPositionAsHome

This home search process is used when the current position is defined as the home position. There are no parameters to set for this process.

8.6 Plus End of Run Only Search Process

Stages.ini file entry: *HomeSearchSequenceType = PlusEndOfRunHomeSearch*

This home search process is used only when the plus end of run signal is used for the home search.

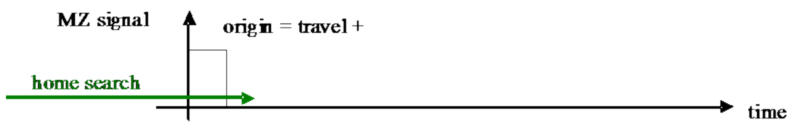


Figure 41: Plus end of run home search process.

8.6.1 Maximum Velocity

Stages.ini file entry: *HomeSearchMaximumVelocity*

The home search maximum velocity, *HomeSearchMaximumVelocity* (units/s), sets the maximum velocity of the profile generator during the home search process. This value must be greater than zero and less than or equal to the *MaximumVelocity* (cf. § 7.2.4).

The default value is one half of the *MaximumVelocity*.

8.6.2 Maximum Acceleration

Stages.ini file entry: *HomeSearchMaximumAcceleration*

The home search maximum acceleration, *HomeSearchMaximumAcceleration* (units/s²), sets the maximum acceleration of the profile generator during the home search process. This value must be greater than zero and less than or equal to the *MaximumAcceleration* (cf. § 7.2.5).

The default value is one half of the maximum acceleration.

8.6.3 Time Out

Stages.ini file entry: *HomeSearchTimeOut*

The home search time out, *HomeSearchTimeOut* (s), sets the time out value for the home search process. When the *HomeSearchTimeOut* time is elapsed and the home search process is not yet finished, the XPS controller will generate an emergency stop and abort the home search process. The *HomeSearchTimeOut* value must be greater than or equal to the profile generator period (400 µs).

8.7 Index Only Search Process

Stages.ini file entry: *HomeSearchSequenceType = IndexHomeSearch*

This home search process is used only when the plus end of run signal is used for the home search.

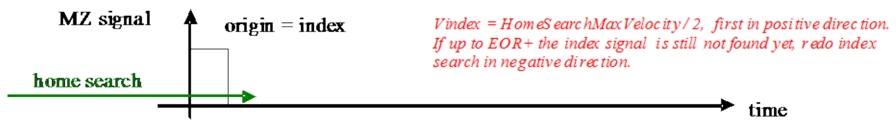

MZ signal
origin = index
Vindex = HomeSearchMaxVelocity / 2, first in positive direction.
If up to EOR+ the index signal is still not found yet, redo index search in negative direction.
home search
time

Figure 42: Index home search process.

8.7.1 Maximum Velocity

Stages.ini file entry: *HomeSearchMaximumVelocity*

The home search maximum velocity, *HomeSearchMaximumVelocity* (units/s), sets the maximum velocity of the profile generator during the home search process. This value must be greater than zero and less than or equal to the *MaximumVelocity* (cf. § 7.2.4). The default value is one half of the *MaximumVelocity*.

The index home search velocity is set to *HomeSearchMaximumVelocity/2.*

8.7.2 Maximum Acceleration

Stages.ini file entry: *HomeSearchMaximumAcceleration*

The home search maximum acceleration, *HomeSearchMaximumAcceleration* (units/s^2), sets the maximum acceleration of the profile generator during the home search process. This value must be greater than zero and less than or equal to the *MaximumAcceleration* (cf. § 7.2.5).

The default value is one half of the maximum acceleration.

8.7.3 Time Out

Stages.ini file entry: *HomeSearchTimeOut*

The home search time out, *HomeSearchTimeOut* (s), sets the time out value for the home search process. When the *HomeSearchTimeOut* time is elapsed and the home search process is not yet finished, the XPS controller will generate an emergency stop and abort the home search process. The *HomeSearchTimeOut* value must be greater than or equal to the profile generator period (400 µs).

9.0 Index

This index bases on the parameter names used in the stages.ini configuration file.

Service Form

Your Local Representative

Tel.: ___________________

Fax:___________________

Name: ______________________________	Return authorization #: ______________________________ *(Please obtain prior to return of item)*
Company:______________________________	
Address:______________________________	Date: ______________________________
Country:______________________________	Phone Number: ______________________________
P.O. Number:______________________________	Fax Number: ______________________________
Item(s) Being Returned:______________________________	
Model#: ______________________________	Serial #: ______________________________

Description: ____

Reasons of return of goods (please list any specific problems): ______________________________

Newport®

Experience | Solutions

Visit Newport Online at:
www.newport.com

North America & Asia
Newport Corporation
1791 Deere Ave.
Irvine, CA 92606, USA

Sales
Tel.: (800) 222-6440
e-mail: sales@newport.com

Technical Support
Tel.: (800) 222-6440
e-mail: tech@newport.com

Service, RMAs & Returns
Tel.: (800) 222-6440
e-mail: service@newport.com

Europe
MICRO-CONTROLE Spectra-Physics S.A.S
9, rue du Bois Sauvage
91055 Évry CEDEX
France

Sales
Tel.: +33 (0)1.60.91.68.68
e-mail: france@newport.com

Technical Support
e-mail: tech_europe@newport.com

Service & Returns
Tel.: +33 (0)2.38.40.51.55